جمهوری اسلامی ایران
سازمان برنامه و بودجه کشور
سازمان نقشه‌برداری کشور

دستورالعمل پایش مقدماتی سرویس‌های مکانی و بستر اجرایی آن‌ها

نگارش ۱/۰

آخرین ویرایش: ۲۰-۰۷-۱۴۰۵

سازمان نقشه‌برداری کشور — http://www.ncc.gov.ir

اداره کل نظارت، کنترل فنی و استانداردها — گروه استانداردسازی

مهر ۱۴۰۵

پیشگفتار

سازمان نقشه‌برداری کشور به‌عنوان مرجع ملی داده‌ها و سرویس‌های مکانی، مجموعه‌ای از سرورهای تخصصی را برای ارائه‌ی سرویس‌های نقشه‌ای و مکانی به کاربران و سامانه‌های ملی در اختیار دارد. در حال حاضر، در این زمینه، بیش از ده سرور در اداره کل سامانه‌های ملی و زیرساخت‌های اطلاعات مکانی فعالیت می‌کنند که پایداری، در دسترس‌بودن و عملکرد صحیح آن‌ها برای تداوم ارائه خدمات حیاتی است. علاوه بر این، زیرساخت داده‌های مکانی کشور دریافت‌کننده وب‌سرویس‌های مکانی از بیش از ششصد دستگاه ملی و استانی است که پایداری و پاسخ‌گویی آن‌ها نیز در استمرار تبادل و ارائه اطلاعات مکانی اهمیت دارد. ازاین‌رو، پایش مستمر وضعیت سرورها و سرویس‌های مکانی به‌عنوان یکی از ارکان اصلی مدیریت زیرساخت‌های سازمان مطرح می‌شود.

این راهنما حاصل طراحی و پیاده‌سازی سامانه‌ی پایش سرورها و سرویس‌های مکانی سازمان با استفاده از ابزارهای متن‌باز Prometheus، Grafana و GeoHealthCheck است. این سه ابزار به‌ترتیب وظیفه‌ی جمع‌آوری داده‌های عملکردی، نمایش بصری وضعیت سیستم‌ها و پایش تخصصی سرویس‌های مکانی مبتنی بر استانداردهای OGC را بر عهده دارند. اجرای ترکیبی آن‌ها بستری مطمئن برای کنترل سلامت، کارایی و پایداری سرویس‌ها فراهم می‌سازد.

در این مستند، تمامی مراحل به‌صورت عملی و گام‌به‌گام اجرا و ثبت شده است؛ از آماده‌سازی سرورهای لینوکسی و نصب جاوا، Tomcat و GeoServer تا راه‌اندازی Node Exporter و Tomcat Exporter روی سرورهای پایش‌شونده و سپس نصب Prometheus، Grafana و GeoHealthCheck بر روی سرور پایش‌کننده. در ادامه، ارتباط میان این سرویس‌ها برقرار و داشبوردهای مدیریتی و نمودارهای تحلیلی برای پایش مستمر منابع و سرویس‌ها ایجاد گردیده تا مدیران بتوانند در هر لحظه وضعیت سلامت و عملکرد سامانه‌ها را مشاهده کنند.

امید است با به‌کارگیری این مستند، گامی مؤثر در جهت افزایش پایداری، بهبود کیفیت خدمات مکانی و ارتقاء بهره‌وری زیرساخت‌های فناوری اطلاعات سازمان نقشه‌برداری کشور برداشته شود.

اداره کل سامانه‌های ملی و زیرساخت‌های اطلاعات مکانی

مهر ۱۴۰۵

جناب آقای پیمان بکتاش - معاون محترم فنی و تولید اطلاعات مکانی
جناب آقای حمیدرضا سیدین بروجنی -معاون محترم فناوری و ارائه اطلاعات مکانی
جناب آقای رامین پاپی - مدیرکل محترم دفتر ریاست ، روابط عمومی و امور بین الملل
جناب آقای علی سلطانپور - مدیر محترم آبنگاری و امور جزر و مدی
سرکار خانم معصومه آمیغ پی - سرپرست محترم اداره کل نقشه برداری زمینی و بنیادی
جناب آقای منصور حسن پور - مدیر کل محترم نقشه برداری هوایی و فضایی
جناب آقای علی مدد - مدیر کل محترم سامانه های ملی و زیر ساخت های اطلاعات مکانی
جناب آقای مهدی سعیدی انجیله - مدیر محترم نقشه نگاری و اطلس های ملی
جناب آقای نعمت الله حبشی - مدیر محترم خدمات فنی
جناب آقای محسن رجب زاده - مدیر محترم مدیریت برنامه ریزی ، بودجه و پژوهش

موضوع : ابلاغ دستورالعمل پایش مقدماتی سرویس‌های مکانی و بستر اجرایی آن‌ها

سلام و علیکم؛

با گسترش زیرساخت‌های فناوری اطلاعات و وابستگی روزافزون سازمان‌ها به سامانه‌های الکترونیکی، ضرورت پایش مستمر وضعیت سرویس‌ها و بستر آن‌ها به‌عنوان یکی از ارکان کلیدی پایداری خدمات، بیش از پیش مورد توجه قرار گرفته است. در بسیاری از فرآیندهای سازمانی، قطع ارتباط کوتاه‌مدت یا طولانی‌مدت یک سرویس یا افت کارایی در یک سرور، می‌تواند موجب توقف عملیات، اختلال در ارائه خدمت، کاهش کیفیت تجربه کاربران و حتی بروز خسارات مالی و فنی شود. به همین دلیل، ایجاد یک نظام یکپارچه برای نظارت، ثبت، تحلیل و مدیریت وضعیت سرورها و سرویس‌های حیاتی، از الزامات مدیریت زیرساخت به شمار می‌آید. در این زمینه با توسعه استانداردهای بین‌المللی در حوزه نظارت و ابزارهای نرم‌افزاری تخصصی، امکان جمع‌آوری داده‌های لحظه‌ای، تحلیل روندهای عملکردی، تشخیص رخدادهای قطعی و ارائه داشبوردهای مدیریتی فراهم شده و مدیریت سرویس‌ها به‌صورت علمی و استاندارد قابل انجام است.

در این خصوص؛ دستورالعمل **"پایش مقدماتی سرویس‌های مکانی و بستر اجرایی آن‌ها"** با هدف ایجاد یک چارچوب منسجم برای نظارت بر عملکرد سرورها و ارائه سرویس‌های حیاتی، از نوع لازم‌الاجرا ابلاغ می‌گردد. امید است با به‌کارگیری این دستورالعمل، گامی مؤثر در جهت افزایش پایداری، بهبود کیفیت خدمات مکانی و ارتقاء بهره‌وری زیرساخت‌های فناوری اطلاعات مکانی در کشور برداشته شود.

سیداسکندر صیدایی

بسمه تعالی

صورتجلسه تایید استاندارد / دستورالعمل سازمان نقشه برداری کشور

عنوان: تائیدیه نسخه دستورالعمل با عنوان

" راهنمای پایش سرورها و سرویس‌های مکانی با استفاده از rafana/Promethus و GeoHealthCheck "

جمع بندی:

- نسخه دستورالعمل با عنوان " راهنمای پایش سرورها و سرویس‌های مکانی با استفاده از Grafana/Promethus و GeoHealthCheck" در تاریخ ۱۴۰۵/۰۷/۰۱ مورد تائید می باشد.

سید عبدالرضا سعادت	حمیدرضا سیدین بروجنی	ساحل قلعه شاهینی
مهدی سعیدی انجیله	عطیه ثقه مجتهدی	مهین حسینی اصل
زهرا کشمیری	میلاد رفیعی طرقی	رقیه فتحی الماس
مهرنوش امتی	زهرا نام پرست	محمدرضا حسینی
رامین پاپی	محمداحمدلو	

طبق صورتجلسه ۲۹ کمیته استاندارد عنوان اصلاح شده دستورالعمل به صورت زیر است:

دستورالعمل پایش مقدماتی سرویس‌های مکانی و بستر اجرایی آن‌ها

تهیه و کنترل « دستورالعمل پایش مقدماتی سرویس‌های مکانی و بستر اجرایی آن‌ها»

اعضای گروه تهیه کننده:

علیرضا امیری (مسئول گروه)	سازمان نقشه‌برداری کشور	کارشناسی‌ارشد سامانه‌های اطلاعات مکانی
مینا مهران‌فر	سازمان نقشه‌برداری کشور	کارشناسی‌ارشد سامانه‌های اطلاعات مکانی و ژئوانفورماتیک
اسماعیل سعادت‌زاده	سازمان نقشه‌برداری کشور	دانشجوی دکتری سامانه‌های اطلاعات مکانی

اعضای گروه نظارت :

مهدی سعیدی انجیله	سازمان نقشه‌برداری کشور	کارشناسی‌ارشد مهندسی فناوری اطلاعات و مدیریت

ویراستار:

سارا مجردی	سازمان نقشه‌برداری کشور	کارشناسی ارشد سنجش از دور

اعضای گروه هدایت و راهبری:

عبدالرضا سعادت	سازمان نقشه‌برداری کشور	دکتری ژئودزی
ساحل قلعه‌شاهینی	سازمان نقشه‌برداری کشور	کارشناسی‌ارشد RS & GIS

فهرست مطالب

فصل اول ۱

۱-۱- انگیزه و ضرورت ۳

۱-۲- امکان‌سنجی اجرایی ۵

۱-۳- مزایای استفاده از محیط لینوکس ۶

۱-۴- ساختار کلی دستورالعمل ۸

فصل دوم ۱۱

۲-۱- لینوکس ۱۳

۲-۱-۱- ساختار لینوکس ۱۳

۲-۱-۲- توزیع‌های لینوکس ۱۴

۲-۱-۳- مدیریت دسترسی‌ها در لینوکس ۱۵

۲-۱-۴- رنگ‌بندی فایل‌ها و ارتباط آن با دسترسی‌ها در لینوکس ۱۷

۲-۱-۵- دستورات پرکاربرد لینوکس ۱۸

۲-۲- مفاهیم پایش و انواع آن ۲۱

۲-۲-۱- پایش عملکرد سیستمی ۲۱

۲-۲-۲- پایش سرویس‌های عمومی ۲۲

۲-۲-۳- پایش سرویس‌های مکانی ۲۲

۲-۳- ابزارهای رایج پایش ۲۳

۲-۳-۱- آشنایی با Prometheus ۲۳

۲-۳-۲- آشنایی با Grafana ۲۶

۲-۳-۳- آشنایی با GeoHealthCheck ۲۹

فصل سوم ۳۳

۳-۱- راه‌اندازی سرویس‌های مکانی ۳۵

۳-۱-۱- نصب Apache Tomcat ۳۶

۳-۱-۲- نصب GeoServer ۳۹

۳-۱-۳- مشاهده سرویس‌های مکانی ۴۲

۳-۱-۴- راه‌اندازی سرویس‌های مکانی از پیش موجود ۴۸

۳-۲- آماده‌سازی سرور پایش‌شونده جهت پایش ۵۲
۳-۲-۱- نصب Node-Exporter ۵۳
۳-۲-۲- نصب Tomcat-Exporter ۵۵
فصل چهارم ۵۹
۴-۱- پایش معیارهای سیستمی ۶۱
۴-۱-۱- نصب Prometheus ۶۱
۴-۱-۲- داشبورد Prometheus ۶۶
۴-۲- بصری‌سازی پایش ۷۰
۴-۲-۱- نصب Grafana ۷۱
۴-۲-۲- داشبورد Grafana ۷۲
۴-۲-۳- اتصال Prometheus به Grafana ۷۵
۴-۳- پایش سرویس‌های مکانی ۸۲
۴-۳-۱- نصب GeoHealthCheck ۸۲
۴-۳-۲- داشبورد GeoHealthCheck ۸۸
فصل پنجم ۹۵
۵-۱- گزارش‌های دریافتی از Prometheus و Grafana ۹۷
۵-۱-۱- گزارش دسترس‌پذیری سرورها ۱۰۱
۵-۱-۲- گزارش فعالیت پردازنده ۱۰۵
۵-۱-۳- گزارش درگیری حافظه ۱۰۶
۵-۱-۴- گزارش وضعیت دیسک ۱۰۷
۵-۱-۵- گزارش شبکه ۱۰۹
۵-۱-۶- گزارش دسترس‌پذیری تامکت ۱۱۱
۵-۲- گزارش‌های دریافتی از GeoHealthCheck ۱۱۲
۵-۲-۱- گزارش دسترس‌پذیری سرویس‌های مکانی ۱۱۲
۵-۲-۲- گزارش پاسخ‌دهی به درخواست GetCapabilities ۱۱۲
۵-۲-۳- گزارش پاسخ‌دهی به درخواست GetResource یا (GetFeature / GetMap) ۱۱۲
پیوست ۱: متن فایل pavement.py ۱۱۳

فهرست اشکال

شکل ۳-۱ – نمونه پیغام خروجی حاصل از دستور ورژن‌یابی در صورت نصب بودن جاوا ۳۷
شکل ۳-۲ - پیغام خروجی حاصل از دستور ورژن‌یابی در صورت نصب نبودن جاوا ۳۸
شکل ۳-۳ - محتوی پوشه‌ی تامکت ۳۹
شکل ۳-۴ - مشاهده و امکان ویرایش پورت اختصاصی ژئوسرور ۴۲
شکل ۳-۵ - نمایی از صفحه ورودی تامکت ۴۳
شکل ۳-۶ - خطای نمایش صفحه تامکت پیش از ویرایش فایل context.xml ۴۴
شکل ۳-۷ - ویرایش فایل context.xml ۴۴
شکل ۳-۸ - تعریف یا ویرایش نام کاربری و رمز عبور برای مدیریت تامکت ۴۵
شکل ۳-۹ - نمایی از صفحه مدیریت تامکت ۴۵
شکل ۳-۱۰ - نمایی از صفحه اولیه ژئوسرور ۴۶
شکل ۳-۱۱ - نمایی از ابزارهای ژئوسرور پس از ورود کاربر به آن ۴۷
شکل ۳-۱۲ - تنظیمات استایل پیش‌فرض لایه‌ها ۵۰
شکل ۴-۱- نمایی از وضعیت تارگت‌ها در Prometheus ۶۷
شکل ۴-۲ – نمونه‌ای از نمودار پرس‌وجو در گراف Prometheus ۶۸
شکل ۴-۳ – نمونه‌ای از هشدار تعریف‌شده در Prometheus ۶۹
شکل ۴-۴ - نمایی از داشبورد Grafana ۷۳
شکل ۴-۵ - مراحل اضافه کردن منابع داده به Grafana ۷۶
شکل ۴-۶ - تنظیمات اتصال Prometheus به Grafana ۷۷
شکل ۴-۷ - ساخت داشبورد جدید در Grafana ۷۸
شکل ۴-۸ - ساخت پنل جدید در داشبورد Grafana ۷۸
شکل ۴-۹ - انتخاب Prometheus به عنوان منبع داده ۷۹
شکل ۴-۱۰ - محیط ساخت عبارت PromQL در قسمت Query ۸۰
شکل ۴-۱۱ - وارد کرد کد PromQL در قسمت Query ۸۰
شکل ۴-۱۲ – گزینه‌های تنظیمات بصری نمودارهای Grafana ۸۱
شکل ۴-۱۳ - نمایی از داشبورد GeoHealthCheck ۸۹
شکل ۴-۱۴ - اضافه نمودن WMS جهت رصد در GeoHealthCheck ۹۰
شکل ۴-۱۵ - وارد نمودن مشخصات سرویس مکانی هدف پایش در GeoHealthCheck ۹۰
شکل ۴-۱۶ - آدرس موردنیاز برای تعریف در پایش برداشت‌شده از ژئوسرور ۹۱

شکل ۴-۱۷ - ویرایش تنظیمات پایش سرویس ۹۲
شکل ۴-۱۸- ابزارهای پایش سرویس مکانی در GeoHealthCheck ۹۲
شکل ۴-۱۹- نمایی از پایش سرویس‌ها در GeoHealthCheck ۹۳
شکل ۵-۱ - تنظیمات داشبورد برای ساخت متغیر ۹۸
شکل ۵-۲ - ایجاد متغیر برای داشبورد Grafana ۹۸
شکل ۵-۳ - تنظیمات تعریف متغیر در داشبورد Grafana ۹۹
شکل ۵-۴- ذخیره داشبورد پس از تعریف متغیر ۱۰۰
شکل ۵-۵ - امکان فیلتر پنل‌ها بر اساس IP ۱۰۰
شکل ۵-۶ - ایجاد و تنظیم پنل گزارش دسترس‌پذیری سرور ۱۰۱
شکل ۵-۷ - تعریف کوئری‌ها و تنظیمات مربوط به استخراج قطعی‌های سرور ۱۰۲
شکل ۵-۸- ترنسفورم تغییر نام فیلدها ۱۰۳
شکل ۵-۹ – ترنسفورم‌ها و تنظیمات استخراج مقاطع قطعی سرور پایش‌شونده ۱۰۴
شکل ۵-۱۰- جدول نمایش مقاطع قطعی سرور پایش‌شونده در پنل Grafana ۱۰۴
شکل ۵-۱۱ - پنل گزارش فعالیت CPU ۱۰۶
شکل ۵-۱۲ - پنل گزارش درگیری RAM ۱۰۷
شکل ۵-۱۳ - پنل گزارش فضای استفاده شده از دیسک ۱۰۸
شکل ۵-۱۴ - پنل گزارش زمان پاسخ‌دهی دیسک ۱۰۹
شکل ۵-۱۵ - بخش اول تنظیمات پنل گزارش ترافیک شبکه ۱۱۰
شکل ۵-۱۶ - بخش دوم تنظیمات پنل گزارش ترافیک شبکه ۱۱۱

فصل اول

مقدمه

۱-۱- انگیزه و ضرورت

گسترش استفاده از داده‌ها و سرویس‌های مکانی در سامانه‌های اطلاعاتی، موجب افزایش وابستگی فعالیت‌های اجرایی، مدیریتی و تصمیم‌گیری به دسترسی مستمر و قابل‌اعتماد به این خدمات شده است. امروزه بخش قابل‌توجهی از اطلاعات مکانی از طریق سرویس‌های تحت وب در اختیار کاربران و سامانه‌های مختلف قرار می‌گیرد و عملکرد صحیح آن‌ها نقش مهمی در تداوم ارائه خدمات دارد. سازمان نقشه‌برداری کشور نیز به‌عنوان مرجع ملی تولید و ارائه اطلاعات مکانی، مجموعه‌ای از داده‌ها و سرویس‌های تخصصی را در بستر زیرساخت‌های فناوری اطلاعات خود ارائه می‌کند. از این‌رو، حفظ پایداری، دسترس‌پذیری و کارایی این زیرساخت‌ها، یکی از الزامات اساسی برای اطمینان از استمرار خدمات مکانی سازمان محسوب می‌شود.

در حال حاضر، بیش از ده سرور تخصصی در اداره‌کل سامانه‌های ملی و زیرساخت‌های اطلاعات مکانی سازمان نقشه‌برداری کشور فعالیت دارند که وظیفه میزبانی و ارائه سرویس‌های مختلف را بر عهده دارند. علاوه بر این، زیرساخت داده‌های مکانی کشور[۱] دریافت‌کننده وب‌سرویس‌های مکانی از بیش از ۶۰۰ دستگاه ملی و استانی است که پایداری و عملکرد صحیح آن‌ها نیز در تداوم ارائه خدمات مکانی اهمیت دارد. از این‌رو، ضرورت پایش علاوه بر زیرساخت‌های داخلی سازمان و واحدهای تابعه آن، سرویس‌های دریافتی از سایر دستگاه‌ها را نیز در بر می‌گیرد و این دستورالعمل می‌تواند در سطح استان‌ها نیز برای پایش سرورها و سرویس‌های دستگاه‌های تابعه توسط متولیان استانی مورد استفاده قرار گیرد. عملکرد این سرورها به عوامل متعددی از جمله وضعیت منابع سخت‌افزاری، سیستم‌عامل، نرم‌افزارهای اجرایی و ارتباطات شبکه وابسته است. بروز اختلال در هر یک از این اجزا می‌تواند موجب کاهش سرعت پاسخ‌گویی، ایجاد خطا در پردازش درخواست‌ها یا قطع کامل دسترسی به سرویس‌ها شود. با توجه به تعدد سرورها و تنوع خدمات ارائه‌شده، بررسی دستی و موردی وضعیت آن‌ها نمی‌تواند به‌تنهایی پاسخ‌گوی نیازهای نظارتی سازمان باشد و استفاده از یک سازوکار منظم برای پایش مستمر عملکرد زیرساخت‌ها ضرورت پیدا می‌کند.

این ضرورت در مورد سرویس‌های مکانی اهمیت بیشتری دارد، زیرا ماهیت داده‌ها و پردازش‌های مرتبط با آن‌ها با بسیاری از سرویس‌های عمومی تفاوت دارد. سرویس‌های مکانی معمولاً با حجم قابل‌توجهی از داده‌های برداری و رستری، لایه‌های اطلاعاتی متعدد و درخواست‌های پردازشی متنوع سر و کار دارند. برای مثال، پاسخ به درخواست نمایش نقشه ممکن است مستلزم خواندن داده‌ها، اعمال سبک‌های نمایشی، پردازش محدوده مکانی موردنظر و تولید تصویر خروجی باشد. همچنین، افزایش تعداد درخواست‌های هم‌زمان یا تغییر محدوده و مقیاس نمایش می‌تواند میزان مصرف منابع پردازشی و حافظه را تحت تأثیر قرار دهد. بنابراین، عملکرد مناسب این سرویس‌ها علاوه بر دسترس‌پذیری سرور، به ظرفیت پردازشی و وضعیت اجزای نرم‌افزاری آن نیز وابسته است.

از سوی دیگر، فعال بودن یک سرور لزوماً به معنای عملکرد صحیح سرویس‌های مستقر بر روی آن نیست. ممکن است سرور از طریق شبکه در دسترس باشد، اما نرم‌افزار ارائه‌دهنده سرویس متوقف شده یا در پاسخ به درخواست کاربران با خطا مواجه شود. حتی

---

[۱] SDI: Spatial Data Infrastructure

در شرایطی که نرم‌افزار سرویس‌دهنده فعال است، احتمال بروز اختلال در اجرای برخی درخواست‌های تخصصی وجود دارد. برای نمونه، یک سرویس نقشه ممکن است به درخواست دریافت اطلاعات توصیفی خود پاسخ دهد، اما هنگام تولید نقشه یا بازیابی داده‌های مکانی با مشکل مواجه شود. در چنین شرایطی، کنترل وضعیت سرور به‌تنهایی برای تشخیص اختلال کافی نخواهد بود و لازم است عملکرد سرویس در سطح درخواست‌های استاندارد نیز مورد بررسی قرار گیرد.

یکی دیگر از چالش‌های مدیریت زیرساخت‌های سازمانی، شناسایی به‌موقع مشکلات و تعیین منشأ بروز آن‌هاست. در نبود یک سامانه پایش متمرکز، شناسایی اختلالات ممکن است به گزارش کاربران یا بررسی‌های موردی کارشناسان وابسته باشد. چنین رویکردی علاوه بر افزایش زمان تشخیص مشکل، دستیابی به اطلاعات دقیق درباره زمان وقوع، مدت تداوم و شرایط مؤثر بر اختلال را دشوار می‌سازد. همچنین، بررسی وضعیت لحظه‌ای سیستم بدون دسترسی به سوابق عملکردی، امکان مقایسه شرایط عادی و غیرعادی را محدود می‌کند. در مقابل، ثبت مستمر داده‌های عملکردی و نتایج بررسی سرویس‌ها می‌تواند اطلاعات موردنیاز برای شناسایی تغییرات نامتعارف و بررسی علل احتمالی اختلالات را فراهم سازد.

پایش مستمر منابع سیستمی از جمله پردازنده، حافظه، فضای ذخیره‌سازی و ارتباطات شبکه، امکان مشاهده تغییرات بار کاری و بررسی وضعیت بهره‌برداری از منابع را فراهم می‌کند. این اطلاعات در کنار وضعیت سرویس‌های نرم‌افزاری و زمان پاسخ‌گویی به درخواست‌ها، می‌توانند تصویری جامع‌تر از عملکرد زیرساخت ارائه دهند. برای مثال، افزایش زمان پاسخ‌دهی یک سرویس مکانی ممکن است با افزایش مصرف پردازنده، کمبود حافظه، تأخیر در عملیات دیسک یا اختلال در ارتباطات شبکه همراه باشد. بررسی هم‌زمان این شاخص‌ها به کارشناسان کمک می‌کند تا ارتباط میان وضعیت منابع و عملکرد سرویس‌ها را ارزیابی کرده و در تشخیص عوامل احتمالی کاهش کارایی، تصمیم‌گیری دقیق‌تری داشته باشند.

علاوه بر تشخیص مشکلات جاری، نگهداری سوابق پایش برای ارزیابی عملکرد زیرساخت در بازه‌های زمانی مختلف اهمیت دارد. ثبت اطلاعات مربوط به دسترس‌پذیری سرورها، مقاطع قطعی، میزان استفاده از منابع و وضعیت پاسخ‌گویی سرویس‌ها، امکان تهیه گزارش‌های دوره‌ای و بررسی روند تغییرات را فراهم می‌سازد. این گزارش‌ها می‌توانند مبنایی برای شناسایی مشکلات تکرارشونده، بررسی تغییرات بار کاری و ارزیابی نیاز به بهبود یا توسعه منابع زیرساختی باشند. همچنین، استفاده از شاخص‌های مشخص و قابل‌اندازه‌گیری، امکان مقایسه وضعیت سرورهای مختلف و ارزیابی نتایج اقدامات اصلاحی را فراهم می‌کند.

از منظر مدیریتی نیز وجود یک بستر یکپارچه برای مشاهده وضعیت سرورها و سرویس‌های مکانی اهمیت ویژه‌ای دارد. پراکندگی اطلاعات عملکردی در محیط‌های مختلف، بررسی هم‌زمان وضعیت زیرساخت‌ها و تهیه گزارش‌های منسجم را دشوار می‌سازد. در مقابل، تجمیع اطلاعات در داشبوردهای نظارتی، امکان مشاهده وضعیت جاری و سوابق عملکردی را در محیطی مشخص فراهم می‌کند. این قابلیت می‌تواند به کاهش زمان موردنیاز برای بررسی وضعیت سامانه‌ها، اولویت‌بندی اقدامات نگهداری و افزایش هماهنگی میان کارشناسان مسئول کمک کند. همچنین، ارائه گزارش‌های قابل‌فهم از وضعیت زیرساخت، زمینه استفاده از اطلاعات فنی در تصمیم‌گیری‌های مدیریتی را فراهم می‌سازد.

با توجه به ساختار اجرایی سازمان نقشه‌برداری کشور، وجود روشی مشخص و قابل‌تکرار برای راه‌اندازی و بهره‌برداری از سامانه پایش نیز ضروری است. استفاده از یک چارچوب مدون موجب می‌شود مراحل آماده‌سازی سرورها، جمع‌آوری اطلاعات عملکردی، بررسی وضعیت سرویس‌ها و تهیه گزارش‌های موردنیاز به شیوه‌ای منظم انجام شود. چنین رویکردی وابستگی فرایند پایش به بررسی‌های پراکنده را کاهش داده و امکان توسعه تدریجی آن به سایر سرورها و سرویس‌های سازمان را فراهم می‌کند. همچنین، مستندسازی مراحل اجرایی و تنظیمات موردنیاز می‌تواند در نگهداری، رفع اشکال و بازسازی بستر پایش مورد استفاده قرار گیرد.

بر همین اساس، ضرورت تدوین این دستورالعمل از نیاز به ایجاد یک سازوکار منسجم برای پایش مقدماتی سرویس‌های مکانی و بستر اجرایی آن‌ها ناشی می‌شود. رویکرد موردنظر، بررسی وضعیت زیرساخت را در سه سطح منابع سیستمی، نرم‌افزارهای ارائه‌دهنده خدمات و عملکرد سرویس‌های مکانی دنبال می‌کند. برای تحقق این هدف، از قابلیت‌های ابزارهای متن‌باز استفاده شده است تا امکان جمع‌آوری و ثبت شاخص‌های عملکردی، نمایش وضعیت سرورها و بررسی پاسخ‌گویی سرویس‌های مکانی فراهم شود. حاصل این رویکرد، ایجاد بستری اولیه برای شناخت بهتر وضعیت زیرساخت، شناسایی اختلالات و پشتیبانی از فرایند نگهداری و مدیریت خدمات مکانی سازمان خواهد بود.

۱-۲- امکان‌سنجی اجرایی

استقرار سامانه پایش سرورها و سرویس‌های مکانی، مستلزم بررسی شرایط فنی زیرساخت موجود و فراهم بودن پیش‌نیازهای سخت‌افزاری، نرم‌افزاری و ارتباطی است. امکان پیاده‌سازی این سامانه به عواملی از جمله نوع سیستم‌عامل سرورها، ظرفیت منابع پردازشی، سطح دسترسی‌های مدیریتی و محدودیت‌های شبکه وابسته است. با توجه به تفاوت زیرساخت‌های فناوری اطلاعات در دستگاه‌های مختلف، لازم است پیش از راه‌اندازی، سازگاری ابزارهای موردنیاز با محیط اجرایی و امکان برقراری ارتباط میان اجزای سامانه بررسی شود.

یکی از ملاحظات اصلی، انتخاب سیستم‌عامل مناسب برای اجرای ابزارهای پایش است. اگرچه در این دستورالعمل از سیستم‌عامل لینوکس استفاده شده است، بهره‌گیری از آن برای تمامی اجزای سامانه الزامی نیست. ابزارهای Prometheus و Grafana قابلیت اجرا در محیط ویندوز را نیز دارند، اما برخی ابزارهای جانبی به سیستم‌عامل وابسته هستند. برای مثال، Node Exporter برای جمع‌آوری اطلاعات عملکردی سرورهای لینوکسی طراحی شده است و در محیط ویندوز می‌توان از Windows Exporter استفاده کرد. همچنین، استقرار GeoHealthCheck عمدتاً در محیط‌های مبتنی بر لینوکس مستند شده و اجرای آن در ویندوز نیازمند فراهم‌سازی محیط اجرایی سازگار است. بر این اساس، استفاده از لینوکس به دلیل سازگاری بیشتر با مجموعه ابزارها و روش‌های اجرایی این دستورالعمل، گزینه مناسب‌تری محسوب می‌شود. بااین‌حال، انتخاب لینوکس برای سرور پایش‌کننده به معنای ضرورت تغییر سیستم‌عامل سرورهای پایش‌شونده نیست.

از نظر معماری اجرایی، سامانه به محیطی برای جمع‌آوری، ذخیره‌سازی و نمایش اطلاعات پایش نیاز دارد که می‌تواند بر روی یک سرور فیزیکی یا ماشین مجازی مستقر شود. استفاده از سرور مستقل برای پایش‌کننده، به‌منظور تفکیک بار پردازشی و جلوگیری

از تأثیر فعالیت‌های پایش بر سرویس‌های اصلی، ترجیح دارد. منابع سخت‌افزاری این سرور باید متناسب با تعداد منابع پایش‌شونده، تناوب جمع‌آوری اطلاعات و مدت نگهداری داده‌ها تعیین شوند. برای استقرار اولیه در مقیاس محدود، چهار هسته پردازشی و هشت گیگابایت حافظه اصلی می‌تواند به‌عنوان نقطه شروع در نظر گرفته شود، اما کفایت این ظرفیت به حجم واقعی داده‌ها و بار پردازشی وابسته است. همچنین، فضای ذخیره‌سازی مناسب برای نگهداری سوابق عملکردی ضروری است.

در سمت سرورهای پایش‌شونده، پیش‌نیازها با توجه به نوع پایش متفاوت خواهند بود. دریافت شاخص‌های سیستمی مستلزم نصب ابزار جمع‌آوری اطلاعات متناسب با سیستم‌عامل است و بررسی عملکرد محیط‌های اجرایی مانند Tomcat نیز به پیکربندی ابزارهای مرتبط نیاز دارد. در مقابل، پایش دسترس‌پذیری و پاسخ‌گویی وب‌سرویس‌های مکانی، لزوماً نیازمند نصب نرم‌افزار بر روی سرور ارائه‌دهنده خدمت نیست و می‌تواند از طریق ارسال درخواست به آدرس سرویس انجام شود. این قابلیت، امکان بررسی سرویس‌های دریافتی از سایر دستگاه‌های ملی و استانی را بدون نیاز به دسترسی مدیریتی به زیرساخت داخلی آن‌ها فراهم می‌کند.

از مهم‌ترین الزامات اجرایی، برقراری ارتباط شبکه‌ای میان سرور پایش‌کننده و منابع موردنظر است. این ارتباط باید امکان دریافت شاخص‌های عملکردی و ارسال درخواست به سرویس‌های مکانی را فراهم سازد. در محیط‌های سازمانی، دسترسی به سرورها و درگاه‌های ارتباطی معمولاً تحت کنترل سیاست‌های امنیتی قرار دارد و مستلزم اخذ مجوز از واحدهای مسئول است. همچنین، نصب ابزارها و تغییر تنظیمات ممکن است به سطح مشخصی از دسترسی مدیریتی نیاز داشته باشد. بنابراین، شناسایی مسیرهای ارتباطی، تعیین مجوزهای لازم و محدودسازی دسترسی به اطلاعات عملکردی، از پیش‌نیازهای استقرار سامانه محسوب می‌شوند.

محدودیت دسترسی به اینترنت نیز از دیگر ملاحظات اجرایی است. در بسیاری از شبکه‌های سازمانی، سرورهای عملیاتی به دلایل امنیتی فاقد دسترسی مستقیم به اینترنت هستند. در چنین شرایطی، نصب ابزارهای پایش از طریق تهیه فایل‌های اجرایی و وابستگی‌های نرم‌افزاری و انتقال آن‌ها به سرور مقصد امکان‌پذیر است. در این روش، بررسی سازگاری نسخه‌ها و کامل بودن بسته‌های موردنیاز اهمیت دارد.

در مجموع، اجرای دستورالعمل حاضر مستلزم فراهم بودن بستر پردازشی مناسب، ابزارهای سازگار با سیستم‌عامل، دسترسی‌های لازم و ارتباط شبکه‌ای مجاز میان اجزای سامانه است. در صورت تحقق این پیش‌نیازها، استقرار سامانه پایش مقدماتی بدون تغییر اساسی در ساختار سرویس‌های موجود امکان‌پذیر خواهد بود. همچنین، این ساختار قابلیت به‌کارگیری در زیرساخت‌های سازمانی مختلف و توسعه تدریجی دامنه پایش در سطوح ملی و استانی را دارد.

۱-۳- مزایای استفاده از محیط لینوکس

انتخاب سیستم‌عامل مناسب، یکی از عوامل مؤثر بر کارایی، پایداری و سهولت مدیریت سامانه‌های پایش در محیط‌های سازمانی است. اگرچه ابزارهای پایش قابلیت اجرا در سیستم‌عامل‌های مختلف از جمله ویندوز و لینوکس را دارند، تفاوت در معماری، مدیریت منابع و سازگاری نرم‌افزاری این محیط‌ها می‌تواند بر پیچیدگی استقرار و نگهداری سامانه تأثیرگذار باشد. لینوکس به دلیل ساختار

انعطاف‌پذیر، امکان اجرای سرویس‌ها با حداقل اجزای غیرضروری و سازگاری گسترده با ابزارهای متن‌باز، بستر مناسبی برای پیاده‌سازی سامانه‌های پایش محسوب می‌شود.

یکی از مزایای لینوکس، امکان نصب و اجرای آن بدون محیط گرافیکی است. در توزیع‌های سروری مانند Ubuntu Server، مدیریت سیستم عمدتاً از طریق خط فرمان انجام می‌شود و نیازی به اجرای مداوم رابط گرافیکی و پردازش‌های وابسته به آن وجود ندارد. این ویژگی در مقایسه با نسخه‌های ویندوز سرور دارای محیط گرافیکی، می‌تواند مصرف حافظه اصلی و پردازنده را کاهش داده و منابع بیشتری را در اختیار سرویس‌های کاربردی قرار دهد. البته ویندوز سرور نیز قابلیت اجرا بدون رابط گرافیکی را از طریق Server Core دارد، اما در لینوکس امکان ایجاد محیط اجرایی حداقلی و مدیریت سرویس‌ها از طریق ابزارهای خط فرمان به‌صورت گسترده فراهم است.

لینوکس همچنین امکانات مناسبی برای اجرای طولانی‌مدت سرویس‌های نرم‌افزاری فراهم می‌کند. سامانه مدیریت سرویس systemd، امکان راه‌اندازی خودکار نرم‌افزارها هنگام روشن شدن سرور، کنترل وضعیت اجرا و راه‌اندازی مجدد سرویس‌های متوقف‌شده را فراهم می‌سازد. توزیع‌های دارای پشتیبانی بلندمدت نیز با ارائه اصلاحات امنیتی و به‌روزرسانی‌های نگهداری، برای محیط‌های عملیاتی مناسب هستند. هرچند ویندوز سرور نیز از قابلیت‌های مشابه برای مدیریت سرویس‌ها برخوردار است، ساختار مدیریتی لینوکس و امکان اعمال بسیاری از تغییرات پیکربندی و به‌روزرسانی‌های نرم‌افزاری بدون راه‌اندازی مجدد سیستم، مدیریت سرویس‌های مستمر را تسهیل می‌کند. البته برخی به‌روزرسانی‌های هسته و اجزای اصلی لینوکس نیز ممکن است به راه‌اندازی مجدد نیاز داشته باشند.

از نظر سازگاری نرم‌افزاری، ابزارهای Prometheus و Grafana دارای نسخه‌های رسمی برای هر دو سیستم‌عامل ویندوز و لینوکس هستند. با این‌حال، برخی ابزارهای جانبی پایش به‌صورت اختصاصی برای محیط لینوکس توسعه یافته‌اند. برای مثال، Node Exporter برای جمع‌آوری شاخص‌های عملکردی سیستم‌های لینوکسی استفاده می‌شود، درحالی‌که در ویندوز باید از Windows Exporter استفاده کرد. همچنین، روش‌های متداول استقرار GeoHealthCheck بر بستر لینوکس و محیط‌های سازگار با آن ارائه شده‌اند. بنابراین، استفاده از لینوکس برای اجرای مجموعه ابزارهای موردنظر در این دستورالعمل، موجب هماهنگی بیشتر میان اجزای سامانه و کاهش نیاز به روش‌های اجرایی متفاوت می‌شود.

پشتیبانی از فناوری‌های مبتنی بر محفظه[۲]، از جمله Docker و Kubernetes، از دیگر مزایای لینوکس است. قابلیت‌های هسته لینوکس مانند Namespaces و Control Groups، امکان جداسازی پردازش‌ها و کنترل مصرف منابع محفظه‌ها را فراهم می‌کنند. در محیط لینوکس، Docker Engine می‌تواند مستقیماً بر بستر سیستم‌عامل اجرا شود، درحالی‌که اجرای محفظه‌های لینوکسی بر روی ویندوز نیازمند یک محیط مبتنی بر هسته لینوکس، مانند ماشین مجازی یا WSL2 است. این تفاوت می‌تواند موجب افزایش اجزای اجرایی و پیچیدگی مدیریت در محیط ویندوز شود. همچنین، لینوکس یکی از بسترهای اصلی اجرای گره‌های Kubernetes است و از این نظر برای توسعه سامانه‌های پایش به زیرساخت‌های مبتنی بر محفظه مناسب است.

---

[۲] Container

از دیگر ویژگی‌های لینوکس، قابلیت مدیریت و خودکارسازی فعالیت‌های اجرایی از طریق ابزارهای خط فرمان و اسکریپت‌نویسی است. ابزارهایی مانند Bash و SSH، امکان مدیریت از راه دور سرورها، اجرای دستورات و اعمال تغییرات پیکربندی را فراهم می‌کنند. همچنین، فعالیت‌های تکرارشونده مانند بررسی وضعیت سرویس‌ها، تهیه نسخه پشتیبان و اجرای وظایف زمان‌بندی‌شده را می‌توان با استفاده از اسکریپت‌ها خودکارسازی کرد. اگرچه قابلیت‌های مشابهی از طریق PowerShell و ابزارهای مدیریتی ویندوز وجود دارد، ساختار فایل‌محور تنظیمات و یکپارچگی ابزارهای خط فرمان در لینوکس، استفاده از روش‌های خودکار و قابل‌تکرار را در مدیریت سرورهای متعدد تسهیل می‌کند.

از منظر امنیت نیز لینوکس سازوکارهای متنوعی برای کنترل دسترسی کاربران، محدودسازی مجوزهای اجرایی و حفاظت از منابع سیستم فراهم می‌کند. ابزارهایی مانند AppArmor و سامانه‌های مدیریت بسته مانند APT، امکان اعمال سیاست‌های امنیتی و مدیریت به‌روزرسانی‌ها را فراهم می‌سازند. همچنین، نصب حداقلی سیستم‌عامل و غیرفعال‌سازی سرویس‌های غیرضروری می‌تواند سطح مواجهه با تهدیدات امنیتی را کاهش دهد. بااین‌حال، امنیت بالاتر، ویژگی ذاتی و تضمین‌شده هیچ سیستم‌عاملی نیست و به نحوه پیکربندی، مدیریت دسترسی‌ها و نگهداری آن بستگی دارد.

در مجموع، اگرچه استقرار سامانه پایش در محیط ویندوز از نظر فنی امکان‌پذیر است، استفاده از لینوکس به دلیل امکان اجرای سبک‌تر در پیکربندی‌های حداقلی، سازگاری مستقیم با ابزارهای موردنظر، پشتیبانی بومی از محفظه‌های لینوکسی و سهولت مدیریت سرویس‌ها، برای پیاده‌سازی این دستورالعمل مناسب‌تر ارزیابی می‌شود. بر همین اساس، توزیع Ubuntu Linux به‌عنوان بستر اجرایی سامانه پایش انتخاب شده است، بدون آنکه استفاده از سایر سیستم‌عامل‌ها در سرورهای پایش‌شونده محدود شود.

۱-٤- ساختار کلی دستورالعمل

دستورالعمل حاضر با هدف ارائه روشی کاربردی و گام‌به‌گام برای راه‌اندازی سامانه پایش سرورها، بسترهای اجرایی و سرویس‌های مکانی تدوین شده است. ساختار این مستند به‌گونه‌ای تنظیم شده که علاوه بر تشریح فرایند کامل استقرار سامانه، امکان استفاده مستقل از بخش‌های مختلف آن نیز متناسب با نیاز و امکانات هر دستگاه فراهم باشد. بر این اساس، مطالب در پنج فصل سازمان‌دهی شده‌اند و از معرفی مفاهیم و پیش‌نیازهای فنی آغاز شده و تا مراحل اجرایی، پیکربندی ابزارها و تهیه گزارش‌های عملکردی ادامه می‌یابند.

فصل اول به بیان انگیزه و ضرورت تدوین دستورالعمل، بررسی امکان‌سنجی اجرایی، پیش‌نیازهای استقرار و دلایل انتخاب محیط لینوکس اختصاص دارد. در این فصل، ملاحظات اصلی پیاده‌سازی سامانه بررسی شده و چارچوب کلی استفاده از دستورالعمل مشخص می‌شود.

فصل دوم مبانی نظری و مفاهیم فنی موردنیاز برای اجرای دستورالعمل را ارائه می‌کند. در این فصل، ابتدا ساختار سیستم‌عامل لینوکس، توزیع‌های متداول، مدیریت دسترسی‌ها و دستورات کاربردی آن معرفی می‌شوند. سپس مفاهیم پایش در سطوح سیستمی،

سرویس‌های عمومی و سرویس‌های مکانی تشریح شده و ابزارهای Prometheus، Grafana و GeoHealthCheck از نظر قابلیت‌ها، کاربردها و ویژگی‌های فنی مورد بررسی قرار می‌گیرند. این فصل با هدف ایجاد شناخت اولیه از محیط اجرایی و ابزارهای مورد استفاده تنظیم شده است.

فصل سوم به راه‌اندازی و آماده‌سازی سرور پایش‌شونده اختصاص دارد. در بخش نخست، مراحل آماده‌سازی محیط اجرایی و نصب و پیکربندی Apache Tomcat و GeoServer برای ارائه سرویس‌های مکانی تشریح می‌شود. همچنین، نحوه دسترسی به سرویس‌های راه‌اندازی‌شده و نمونه‌ای از پیاده‌سازی آن‌ها در سازمان نقشه‌برداری کشور ارائه می‌گردد. در بخش دوم، مراحل نصب و تنظیم ابزارهای Node Exporter و Tomcat Exporter به‌منظور فراهم‌سازی امکان دریافت شاخص‌های عملکردی سرور و محیط اجرای سرویس‌ها بررسی می‌شود.

فصل چهارم به استقرار سرور پایش‌کننده و پیکربندی ابزارهای مرکزی پایش اختصاص یافته است. در این فصل، مراحل نصب و راه‌اندازی Prometheus، ایجاد ارتباط با منابع جمع‌آوری اطلاعات و استفاده از Grafana برای نمایش و تحلیل شاخص‌های عملکردی تشریح می‌شود. همچنین، نحوه نصب و پیکربندی GeoHealthCheck و تعریف سرویس‌های مکانی برای بررسی دسترس‌پذیری و پاسخ‌گویی آن‌ها ارائه می‌گردد.

تفکیک مراحل اجرایی در فصل‌های سوم و چهارم با توجه به تفاوت نیازهای دستگاه‌ها و مسئولیت آن‌ها در ارائه یا پایش خدمات صورت گرفته است. برخی دستگاه‌ها ممکن است تنها به آماده‌سازی سرورهای خود برای قرارگیری تحت نظارت یک سامانه پایش موجود نیاز داشته باشند. در مقابل، دستگاه‌هایی که قصد راه‌اندازی سامانه پایش‌کننده را دارند، می‌توانند از مراحل فصل چهارم برای نظارت بر سرورها و سرویس‌های موجود استفاده کنند. بنابراین، اجرای تمامی مراحل فصل سوم، از جمله نصب مجدد سرویس‌دهنده مکانی، پیش‌شرط راه‌اندازی سامانه پایش‌کننده نیست. بااین‌حال، دریافت شاخص‌های داخلی سرورها مستلزم فراهم بودن ابزارهای جمع‌آوری اطلاعات و دسترسی‌های مربوط است، درحالی‌که بررسی پاسخ‌گویی وب‌سرویس‌های مکانی می‌تواند بدون نصب ابزار بر روی سرور مبدأ انجام شود. این تفکیک، امکان بهره‌گیری از دستورالعمل را متناسب با شرایط زیرساختی و سطح دسترسی دستگاه‌های مختلف فراهم می‌کند.

فصل پنجم به تهیه گزارش‌ها و نمایش نتایج حاصل از پایش اختصاص دارد. در این فصل، نحوه ایجاد داشبوردها و گزارش‌های مرتبط با دسترس‌پذیری سرورها، فعالیت پردازنده، مصرف حافظه، وضعیت دیسک، ارتباطات شبکه و دسترس‌پذیری Tomcat بررسی می‌شود. همچنین، گزارش‌های مربوط به وضعیت سرویس‌های مکانی و نتایج بررسی درخواست‌های استاندارد از طریق GeoHealthCheck ارائه می‌گردد. در پیوست دستورالعمل نیز متن فایل مورد استفاده برای نصب و آماده‌سازی محیط اجرایی GeoHealthCheck درج شده است.

در مجموع، ترتیب فصل‌ها بر اساس روند منطقی آشنایی با مفاهیم، آماده‌سازی بستر اجرایی، استقرار ابزارهای پایش و بهره‌برداری از نتایج تنظیم شده است. بااین‌حال، ساختار تفکیک‌شده بخش‌های اجرایی این امکان را فراهم می‌کند که دستگاه‌های ملی و استانی،

با توجه به نقش خود در ارائه یا دریافت خدمات مکانی، از قسمت‌های مرتبط دستورالعمل استفاده کرده و در صورت نیاز، سایر اجزای سامانه پایش را به زیرساخت موجود اضافه کنند.

فصل دوم

آشنایی با مفاهیم

۲-۱- لینوکس

لینوکس یک سیستم عامل متن‌باز[۳] است که در سال ۱۹۹۱ میلادی توسط لینوس توروالدز[۴] توسعه یافت. این سیستم عامل بر پایه مفاهیم سیستم عامل یونیکس طراحی شده و هدف آن ارائه یک محیط پایدار، امن و قابل انعطاف برای کاربران و برنامه‌نویسان بود. یکی از ویژگی‌های مهم لینوکس متن‌باز بودن آن است؛ به این معنی که کد منبع هسته[۵] و بسیاری از نرم‌افزارهای آن در دسترس عموم قرار دارند و هر کسی می‌تواند آن را مطالعه، تغییر یا توزیع کند.

لینوکس به دلیل انعطاف‌پذیری و امنیت بالا، در زمینه‌های مختلف از جمله سرورها، کامپیوترهای شخصی، دستگاه‌های تلفن همراه، سوپرکامپیوترها و اینترنت اشیاء[۶] استفاده می‌شود. همچنین، بسیاری از وب‌سایت‌های بزرگ دنیا مانند گوگل، فیس‌بوک و آمازون برای سرورهای خود از توزیع‌های لینوکس بهره می‌برند.

یکی دیگر از ویژگی‌های مهم لینوکس، پایداری و مدیریت منابع سیستم است. لینوکس می‌تواند بدون نیاز به ریست طولانی مدت، برای ماه‌ها یا حتی سال‌ها بدون مشکل به کار خود ادامه دهد. به همین دلیل، این سیستم عامل در محیط‌های سروری و صنعتی بسیار محبوب است.

۲-۱-۱- ساختار لینوکس

لینوکس از چندین بخش اصلی تشکیل شده است که هر کدام نقش مهمی در عملکرد سیستم دارند.

الف) هسته (Kernel)

هسته، قلب سیستم عامل لینوکس است و وظایف زیر را بر عهده دارد:

- مدیریت پردازش‌ها و اجرای برنامه‌ها
- کنترل حافظه اصلی و مجازی
- ارتباط با سخت‌افزار مانند کارت گرافیک، دیسک‌ها و پرینترها
- مدیریت دستگاه‌های جانبی و درایورها
- تامین امنیت و مجوز دسترسی‌ها

---

[۳] Open Source

[۴] Linus Torvalds

[۵] Kernel

[۶] IoT (Internet of Things)

هسته لینوکس یک هسته یکپارچه[۷] است که به صورت مدولار طراحی شده است. این طراحی به سیستم اجازه می‌دهد تا ماژول‌ها و بخش‌های اضافی هسته را به صورت پویا بارگذاری یا حذف کند، بدون اینکه نیاز به ریست کامل سیستم باشد.

ب) فضای کاربری (User Space)

فضای کاربری بخشی است که کاربران و برنامه‌ها با آن تعامل دارند. این فضا شامل:

- نرم‌افزارهای کاربردی مانند مرورگر وب، ویرایشگر متن و برنامه‌های گرافیکی،
- کتابخانه‌ها[۸] که توابع و امکاناتی را در اختیار برنامه‌ها قرار می‌دهند،
- شل[۹] که رابطی بین کاربر و هسته است و دستورات متنی را اجرا می‌کند.

شل‌های معروف لینوکس شامل Bash، Zsh و Fish هستند. شل به کاربران امکان اجرای دستورات، اسکریپت‌نویسی و مدیریت فایل‌ها را می‌دهد.

ج) سیستم فایل (File System)

لینوکس از سیستم فایل سلسله‌مراتبی استفاده می‌کند. در این سیستم، تمام فایل‌ها و دایرکتوری‌ها زیر یک دایرکتوری ریشه[۱۰] قرار دارند. مهم‌ترین دایرکتوری‌ها عبارتند از:

/ home: محل فایل‌های کاربران
/ etc: تنظیمات و پیکربندی سیستم
/ var: فایل‌های متغیر و لاگ‌ها
/ usr: نرم‌افزارها و کتابخانه‌ها
/bin و / sbin: دستورات پایه و اجرایی

هر فایل و دایرکتوری دارای مالک، گروه و مجوزهای دسترسی است که امنیت و کنترل منابع را تضمین می‌کند.

۲-۱-۲- توزیع‌های لینوکس

لینوکس به دلیل متن‌باز بودن، توسط شرکت‌ها و جوامع مختلف به شکل توزیع[۱۱] ارائه می‌شود. هر توزیع مجموعه‌ای از هسته، نرم‌افزارها و ابزارهای مدیریتی را شامل می‌شود و برای هدف خاصی طراحی شده است. معروف‌ترین توزیع‌ها:

Ubuntu: مناسب برای مبتدیان، دارای رابط کاربری گرافیکی ساده و جامعه پشتیبانی بزرگ. مناسب برای دسکتاپ و سرور.

---

[۷] Monolithic

[۸] Libraries

[۹] Shell

[۱۰] Root

[۱۱] Distribution

Debian: توزیع پایدار و امن، مناسب برای سرورهای سازمانی. در این توزیع به‌روزرسانی‌ها با دقت انتخاب می‌شوند.

CentOS / Rocky Linux: توزیع‌هایی مشابه Red Hat Enterprise Linux (RHEL) با پایداری بالا، مناسب برای سرورهای سازمانی و محیط‌های تولید.

Fedora: به‌روز و پیشرفته، مناسب برای توسعه‌دهندگان و کسانی که می‌خواهند آخرین ویژگی‌ها را تجربه کنند.

Arch Linux: سبک و انعطاف‌پذیر، مناسب برای کاربران حرفه‌ای که می‌خواهند سیستم را شخصی‌سازی کنند.

هر توزیع دارای مدیریت بسته‌های نرم‌افزاری[۱۲] مخصوص به خود است:

- Debian و Ubuntu از APT[۱۳] استفاده می‌کنند؛
- Red Hat، CentOS و Fedora از YUM[۱۴] / DNF[۱۵] استفاده می‌کنند؛
- Arch Linux از Pacman[۱۶] استفاده می‌کند.

هر توزیع لینوکس دارای نسخه[۱۷]های متفاوتی است که بسته به ویژگی‌ها و به‌روزرسانی‌ها منتشر می‌شوند. برخی نسخه‌ها LTS (Long Term Support) می‌باشند که پشتیبانی طولانی‌مدت دارند و مناسب محیط‌های کاری حساس هستند. به عنوان مثال:

- Ubuntu 22.4 LTS: پشتیبانی تا ۵ سال
- Ubuntu 23.10: پشتیبانی کوتاه‌مدت برای تجربه امکانات جدید

نسخه‌ها معمولاً شامل به‌روزرسانی‌های امنیتی، اصلاحات باگ و امکانات جدید نرم‌افزاری هستند.

لازم به ذکر است که نسخه مورد استفاده در سرورهای سازمان نقشه‌برداری کشور Ubuntu X LTS می‌باشد.

۲-۱-۳- مدیریت دسترسی‌ها در لینوکس

سیستم دسترسی لینوکس یکی از نقاط قوت آن است؛ بطوریکه این سیستم عامل یکی از قوی‌ترین ویژگی‌های امنیتی خود را در مدیریت دسترسی‌ها و کنترل مجوزهای فایل‌ها و دایرکتوری‌ها نشان می‌دهد. این قابلیت باعث می‌شود کاربران مختلف با سطوح متفاوت، بتوانند منابع سیستم را به شکل امن و مدیریت‌شده استفاده کنند. هر فایل و دایرکتوری دارای:

- مالک (Owner)؛ کاربری که فایل را ایجاد کرده،

---

[۱۲] Package Manager

[۱۳] Advanced Package Tool

[۱۴] Yellowdog Updater, Modified

[۱۵] Dandified YUM

[۱۶] Package Manager

[۱۷] Version

- گروه (Group)؛ مجموعه‌ای از کاربران با دسترسی مشابه،
- سایر کاربران (Others)؛ تمامی کاربران دیگر می‌باشد.

برای کنترل دسترسی، سه نوع دسترسی اصلی وجود دارد:

- خواندن (Read / r) مشاهده محتویات فایل یا لیست دایرکتوری
- نوشتن (Write / w): تغییر، ایجاد یا حذف محتویات
- اجرا (Execute / x): اجرای فایل‌های برنامه یا دسترسی به دایرکتوری

مجوزها در لینوکس به صورت سلسله‌مراتبی و ترکیبی اعمال می‌شوند. به این صورت که هر فایل یا دایرکتوری سه سطح مجوز دارد: سطح مالک، سطح گروه و سطح سایر کاربران. این سیستم اجازه می‌دهد که دسترسی‌ها به صورت دقیق و کنترل‌شده اعمال شوند و امنیت سیستم حفظ گردد. به عنوان مثال، یک فایل حساس می‌تواند به گونه‌ای تنظیم شود که تنها مالک آن بتواند فایل را ویرایش کند، اعضای گروه بتوانند فقط محتویات آن را مشاهده کنند و سایر کاربران هیچ دسترسی‌ای به آن نداشته باشند. دستورات اصلی مدیریت دسترسی عبارتند از:

chmod: تغییر مجوزهای فایل

chown: تغییر مالک فایل

chgrp: تغییر گروه فایل

با ترکیب این مجوزها، می‌توان امنیت سیستم را تضمین کرد و کنترل دقیقی روی دسترسی کاربران داشت.

از لحاظ مفهومی، سیستم دسترسی لینوکس با مدل امنیتی مبتنی بر مالکیت و گروه شباهت‌هایی با مدل امنیتی یونیکس دارد، اما با اضافه شدن قابلیت بارگذاری ماژول‌ها و مدیریت مجوزهای پیچیده، انعطاف و کنترل بیشتری ارائه می‌دهد. این سیستم اجازه می‌دهد که امنیت فایل‌ها و دایرکتوری‌ها بدون نیاز به تغییر ساختار اصلی سیستم عامل و بدون تداخل با دیگر کاربران حفظ شود.

علاوه بر این، مدیریت دسترسی‌ها در لینوکس نه تنها شامل فایل‌ها و دایرکتوری‌ها می‌شود، بلکه شامل فرآیندها، منابع سیستم و دسترسی به دستگاه‌ها نیز هست. هر فرآیند در سیستم با یک کاربر و گروه مرتبط است و مجوزهای آن تعیین می‌کند که چه عملیاتی می‌تواند روی منابع سیستم انجام دهد. به همین دلیل، لینوکس از نظر امنیتی بسیار پایدار و مقاوم است و حتی در محیط‌های سروری با تعداد بالای کاربران و فرآیندها، توانایی کنترل کامل دسترسی‌ها را دارد.

۲-۱-۴- رنگ‌بندی فایل‌ها و ارتباط آن با دسترسی‌ها در لینوکس

یکی از ویژگی‌های ترمینال در لینوکس این است که هنگام استفاده از دستور ls --color یا ls -l، فایل‌ها و پوشه‌ها با رنگ‌های متفاوتی نمایش داده می‌شوند. این رنگ‌ها نشان‌دهنده نوع فایل و در بعضی موارد مجوزهای دسترسی[۱۸] هستند. هدف از این رنگ‌بندی، کمک به کاربر برای تشخیص سریع‌تر وضعیت فایل‌ها است.

این رنگ‌ها به طور مستقیم توسط سیستم‌عامل ذخیره نمی‌شوند، بلکه توسط دستور ls --color و متغیر محیطی LS_COLORS تعیین و نمایش داده می‌شوند. به همین دلیل ممکن است در توزیع‌های مختلف لینوکس یا محیط‌های ترمینال متفاوت، کمی تغییر داشته باشند.

رنگ آبی (Blue)

فایل‌ها یا پوشه‌هایی که دایرکتوری[۱۹] هستند، معمولاً به رنگ آبی نمایش داده می‌شوند. این رنگ کمک می‌کند که پوشه‌ها به راحتی از فایل‌های معمولی قابل تشخیص باشند.

رنگ سبز (Green)

فایل‌هایی که دارای دسترسی اجرا (x) هستند، به رنگ سبز نمایش داده می‌شوند. این یعنی فایل قابلیت اجرای مستقیم دارد و می‌تواند به عنوان یک برنامه یا اسکریپت استفاده شود.

رنگ فیروزه‌ای یا آبی روشن (Cyan)

فایل‌هایی که لینک نمادین[۲۰] هستند به رنگ فیروزه‌ای نمایش داده می‌شوند. این رنگ نشان می‌دهد که فایل یک اشاره[۲۱] به فایل یا پوشه دیگری است. اگر مقصد لینک وجود نداشته باشد، معمولاً به رنگ قرمز یا قرمز چشمک‌زن نمایش داده می‌شود.

رنگ قرمز (Red)

این رنگ معمولاً برای فایل‌های فشرده یا آرشیوی (مانند .zip, .tar, .gz) استفاده می‌شود. در بعضی تنظیمات، اگر فایلی دسترسی خطرناک داشته باشد (مثلاً برای همه کاربران world-writable باشد)، به رنگ قرمز یا با پس‌زمینه هشداردهنده نمایش داده می‌شود.

رنگ سفید یا خاکستری (White/Gray)

فایل‌های معمولی که نه پوشه‌اند و نه اجرایی، معمولاً سفید یا خاکستری نمایش داده می‌شوند.

---

[۱۸] permissions

[۱۹] Directory

[۲۰] Symbolic Link

[۲۱] pointer

رنگ زرد (Yellow)

رنگ زرد یا زرد با پس‌زمینه مشکی برای فایل‌های ویژه (Special Files) مانند فایل‌های دستگاه (device files) استفاده می‌شود. این فایل‌ها در دایرکتوری /dev بیشتر دیده می‌شوند (مانند کارت شبکه، دیسک سخت، ترمینال‌ها).

به این ترتیب مثلا وقتی بعد از اجرای chmod 755 file رنگ فایل سبز می‌شود، معنایش این است که آن فایل اکنون قابل اجرا شده است. همچنین اگر پوشه‌ای به رنگ آبی دیده شود، نشان‌دهنده یک دایرکتوری است.

۲-۱-۵- دستورات پرکاربرد لینوکس

در این بخش، مهم‌ترین دستورات لینوکس و کاربرد آن‌ها آورده شده است:

۱) دستورات پایه مدیریت فایل و پوشه‌ها

```
pwd                          # نمایش مسیر جاری
ls                           # نمایش لیست فایل‌ها و پوشه‌ها در مسیر جاری
ls -l                        # نمایش لیست با جزئیات کامل
ls -lh                       # نمایش جزئیات همراه با اندازه فایل‌ها به صورت خوانا (KB, MB)
ls -a                        # نمایش همه فایل‌ها شامل فایل‌های مخفی
cd folder                    # تغییر مسیر به پوشه folder
cd ..                        # برگشت به پوشه والد
cd ~                         # رفتن به پوشه خانه (Home Directory) کاربر جاری
cd                           # برگشت به پوشه خانه کاربر
mkdir folder                 # ایجاد پوشه جدید به نام folder
mkdir -p folder/subfolder    # ایجاد پوشه همراه با زیرپوشه (اگر وجود نداشت)
rmdir folder                 # حذف پوشه خالی
rm file                      # حذف فایل
rm -r folder                 # حذف پوشه همراه با محتویات آن
touch file                   # ایجاد فایل خالی
mv file folder/              # جابجایی فایل از مسیر مبدأ به مقصد (پوشه folder)
mv folder folder_new         # تغییر نام پوشه
cp file folder/              # کپی فایل به مقصد
cp -r folder folder_backup   # کپی پوشه به مقصد به همراه زیرپوشه‌ها
```

۲) نمایش و بررسی محتوای فایل‌ها

```
cat file                     # نمایش محتوای کامل فایل
less file                    # نمایش محتوای فایل صفحه به صفحه
head file                    # نمایش ۱۰ خط اول فایل
head -20 file                # نمایش ۲۰ خط اول فایل
tail file                    # نمایش ۱۰ خط آخر فایل
```

```
tail -f file                    # نمایش زنده آخرین خطوط فایل (کاربردی برای لاگ‌ها)
echo "Hello Linux" > file       # نوشتن متن در فایل (بازنویسی محتوا)
echo "New Line" >> file         # اضافه‌کردن خط جدید به انتهای فایل
nano file                       # ویرایش فایل با ویرایشگر nano
vim file                        # ویرایش فایل با ویرایشگر vim
wc -l file                      # شمارش تعداد خطوط فایل
wc -w file                      # شمارش تعداد کلمات فایل
wc -c file                      # شمارش تعداد کاراکترهای فایل
```

۳) جستجو و فیلتر

```
find . -name file               # جستجوی فایل با نام file در مسیر جاری
find / -type d -name folder     # جستجوی پوشه با نام folder در کل سیستم
grep "text" file                # جستجوی متن "text" در فایل
grep -i "text" file             # جستجوی متن بدون توجه به بزرگی/کوچکی حروف
grep -r "error" folder/         # جستجوی عبارت error در تمام فایل‌های پوشه
grep -n "error" file            # نمایش شماره خط هنگام یافتن متن
cut -d':' -f1 /etc/passwd       # جدا کردن ستون اول فایل با جداکننده ":"
sort file                       # مرتب‌سازی خطوط فایل
uniq file                       # حذف خطوط تکراری در فایل
diff file1 file2                # مقایسه محتوای دو فایل
cmp file1 file2                 # مقایسه باینری دو فایل
```

۴) پردازش‌ها و سرویس‌های کاربردی

```
ps aux                          # نمایش همه پردازش‌های در حال اجرا
top                             # نمایش پردازش‌های فعال به صورت زنده
htop                            # نمایش پردازش‌ها با رابط تعاملی (اگر نصب باشد)
kill -9 PID                     # متوقف کردن پردازش با شناسه (PID)
systemctl status App            # بررسی وضعیت سرویس
systemctl restart App           # راه‌اندازی مجدد سرویس
journalctl -u App -f            # نمایش زنده لاگ‌های سرویس
```

۵) شبکه و مانیتورینگ

```
ping 8.8.8.8                    # تست اتصال به اینترنت (گوگل DNS)
ping server-ip                  # تست اتصال به یک سرور مشخص
curl http://localhost:Port      # بررسی سرویس روی پورت
netstat -tulnp                  # نمایش پورت‌های باز و پردازش‌های استفاده‌کننده
ss -tulnp                       # نمایش دقیق‌تر پورت‌های در حال استفاده
ifconfig                        # نمایش تنظیمات کارت شبکه (در برخی سیستم‌ها نیاز به نصب net-tools)
ip a                            # نمایش آدرس‌های IP کارت‌های شبکه
traceroute example.com          # نمایش مسیر بسته‌ها تا مقصد
```

```
nslookup example.com        # بررسی رکورد DNS برای دامنه
dig example.com             # نمایش اطلاعات کامل DNS دامنه
```

۶) مدیریت بسته‌ها

```
apt update                  # بروزرسانی لیست بسته‌ها (Debian/Ubuntu)
apt upgrade                 # بروزرسانی همه بسته‌های نصب شده
apt install package         # نصب بسته جدید
apt remove package          # حذف بسته
apt purge package           # حذف کامل بسته به همراه تنظیمات
yum update                  # بروزرسانی بسته‌ها (RedHat/CentOS)
dnf install package         # نصب بسته در سیستم‌های جدیدتر RedHat/CentOS
dnf remove package          # حذف بسته در سیستم‌های جدیدتر
pacman -Syu                 # بروزرسانی کامل سیستم (Arch Linux)
pacman -S package           # نصب بسته در Arch Linux
```

۷) مدیریت دسترسی و کاربران

```
ls -l file                  # نمایش مجوزهای فایل
chmod 755 file              # تغییر مجوز فایل (مالک rwx، گروه r-x، سایر r-x)
chmod u+x file              # اضافه کردن مجوز اجرا فقط برای مالک
chmod g-w file              # حذف مجوز نوشتن از گروه
chmod o-r file              # حذف مجوز خواندن از سایر کاربران
chown user file             # تغییر مالک فایل به user
chown user:group file       # تغییر مالک و گروه فایل
chgrp group file            # تغییر گروه فایل
id user                     # نمایش UID و GID کاربر
whoami                      # نمایش نام کاربر جاری
groups                      # نمایش گروه‌هایی که کاربر عضو آن‌هاست
adduser user                # ایجاد کاربر جدید (Debian/Ubuntu)
passwd user                 # تغییر رمز عبور کاربر
deluser user                # حذف کاربر (Debian/Ubuntu)
usermod -aG group user      # اضافه کردن کاربر به یک گروه
sudo command[۳۲]            # اجرای دستور با دسترسی مدیر سیستم (root)
```

لازم به‌ذکر است که اجرای برخی از دستورات، نیاز به دسترسی به اینترنت داشته که در اغلب سرورهای سازمانی به لحاظ مسائل امنیتی این دسترسی قطع شده است.

[۳۲] تنظیمات sudo در فایل /etc/sudoers انجام می‌شود. این فایل مشخص می‌کند که کدام کاربران و گروه‌ها اجازه دارند از sudo استفاده کنند و چه دستورات خاصی را می‌توانند اجرا کنند.

۲-۲- مفاهیم پایش و انواع آن

پایش یا مانیتورینگ[۲۳] سرورها یکی از حیاتی‌ترین وظایف در مدیریت زیرساخت‌های فناوری اطلاعات است. پایش به معنای جمع‌آوری، تحلیل و نمایش مداوم اطلاعات درباره وضعیت منابع سیستم و سرویس‌های در حال اجرا است.

هدف اصلی از پایش سرورها این است که مدیر سیستم بتواند در سریع‌ترین زمان ممکن از مشکلات آگاه شود و قبل از اینکه این مشکلات به خرابی یا توقف سرویس منجر شوند، اقدامات لازم را انجام دهد.

در یک نگاه کلی، پایش را می‌توان به دو بخش اصلی تقسیم کرد:

- پایش سیستمی: که به منابع سخت‌افزاری و نرم‌افزاری پایه‌ی سرور توجه دارد.
- پایش سرویس‌ها: که وضعیت سرویس‌های نرم‌افزاری نصب‌شده روی سرور را بررسی می‌کند.

علاوه بر این‌ها، در دنیای امروز که سرویس‌های مکانی اهمیت زیادی پیدا کرده‌اند، پایش سرویس‌های مکانی نیز به‌عنوان یک دسته‌بندی مهم مطرح است. در تمام این سطوح، جمع‌آوری داده‌های امنیتی، لاگ‌ها و وضعیت منابع، نقش حیاتی دارند.

۲-۲-۱- پایش عملکرد سیستمی

پایش عملکرد سیستم اولین و ابتدایی‌ترین سطح مانیتورینگ سرورها است. هدف از آن اطمینان یافتن از این است که سخت‌افزار و سیستم‌عامل به درستی عمل می‌کنند و منابع در حد بهینه مصرف می‌شوند. مهم‌ترین بخش‌های پایش سیستمی شامل موارد زیر هستند:

۱. پردازنده (CPU): پایش پردازنده به معنی بررسی بار کاری آن است. دستورات پایه‌ای مانند top یا نسخه پیشرفته‌تر آن یعنی htop می‌توانند در لحظه بار پردازنده، پردازش‌های پرمصرف و بار روی هر هسته پردازشی را نمایش دهند. این پایش به ویژه در سرویس‌های مکانی که پردازش‌های سنگین دارند اهمیت زیادی دارد.

۲. حافظه (RAM): وضعیت حافظه اصلی (RAM) نقش اساسی در عملکرد سیستم دارد. کمبود حافظه منجر به استفاده از حافظه مبادله‌ای[۲۴] و در نتیجه کاهش شدید کارآیی می‌شود. ابزارهایی مانند free -h و vmstat برای این سنجش استفاده می‌شوند.

۳. دیسک‌ها و ذخیره‌سازی: سرورها برای ذخیره داده‌ها به دیسک متکی هستند. بررسی فضای باقی‌مانده (df –h) و پایش سرعت ورودی/خروجی دیسک با دستوراتی مانند iostat اهمیت زیادی دارد. در سرویس‌های مکانی که فایل‌های بزرگ نقشه یا پایگاه داده‌های مکانی ذخیره می‌شوند، این بخش حیاتی است.

---

[۲۳] Monitoring

[۲۴] SWAP: به فضایی در هارد دیسک اشاره دارد که برای ذخیره‌سازی موقت داده‌ها و فرآیندهای در حال اجرا در نظر گرفته شده است. هنگامی که حافظه اصلی RAM پر می‌شود، سیستم‌عامل می‌تواند داده‌ها و فرآیندهایی که به‌طور موقت به آن‌ها نیاز ندارد را به فضای SWAP منتقل کند.

۴. شبکه: ارتباط پایدار شبکه اساس عملکرد بیشتر سرویس‌هاست. ابزارهایی مانند iftop، netstat یا ss به مدیر سیستم کمک می‌کنند تا ترافیک شبکه، اتصالات باز و مصرف پهنای باند را پایش کنند. این داده‌ها همچنین برای تشخیص رفتارهای مشکوک امنیتی و تحلیل بار شبکه ضروری هستند.

۵. لاگ‌ها: سیستم‌عامل لینوکس تمامی رخدادهای مهم را در فایل‌های لاگ ذخیره می‌کند. مرور این لاگ‌ها با ابزارهایی مانند journalctl یا بررسی /var/log/ بخش مهمی از پایش سیستمی است. لاگ‌ها اولین منبع برای کشف مشکلات سخت‌افزاری و نرم‌افزاری هستند. همچنین پایش امنیتی شامل شناسایی تلاش‌های ناموفق ورود، بررسی اتصالات مشکوک و کنترل تغییرات غیرمجاز است. ابزارهایی مانند Fail2ban یا IDS/IPSها نقش کلیدی در این بخش دارند.

۲-۲-۲- پایش سرویس‌های عمومی

وقتی منابع پایه سیستم پایدار باشند، نوبت به سرویس‌هایی می‌رسد که روی سرور اجرا می‌شوند. پایش سرویس‌ها یعنی اطمینان از اینکه برنامه‌ها و سرویس‌های نرم‌افزاری نصب‌شده روی سرور بدون خطا و بی‌وقفه کار می‌کنند.

۱. سرویس‌های سیستمی: در لینوکس سرویس‌ها معمولاً با systemd مدیریت می‌شوند. دستور systemctl status <service> وضعیت یک سرویس را مشخص می‌کند. پایش سرویس‌های پایه مانند وب‌سرور (Apache, Nginx)، پایگاه داده (PostgreSQL, MySQL) یا سرویس‌های زمان (NTP) از اهمیت ویژه برخوردار است.

۲. سرویس‌های تخصصی: برای سرویس‌هایی که در سطح سازمانی اجرا می‌شوند (مانند ژئوسرور برای نقشه‌ها، Grafana برای داشبوردها یا Prometheus برای متریک‌ها)، پایش دائمی لازم است. این پایش باید شامل بررسی وضعیت اجرا، مصرف منابع و لاگ‌های خطا باشد.

۳. اعلان‌ها و هشدارها: پایش سرویس‌ها تنها به مشاهده وضعیت فعلی محدود نمی‌شود. ابزارهای پایش باید توانایی تولید هشدار[۲۵] داشته باشند. به‌عنوان مثال، اگر وب‌سرور خاموش شود یا پایگاه داده پاسخگو نباشد، سیستم پایش باید بلافاصله پیام هشدار برای مدیر سیستم ارسال کند. ابزارهایی همانند Grafana به همراه Prometheus امکان ساخت داشبورد و اعلان بصری و ایمیلی را فراهم می‌کنند.

۲-۲-۳- پایش سرویس‌های مکانی

سرویس‌های مکانی به دلیل ماهیت داده‌های حجیم و درخواست‌های پیچیده، نیازمند پایش ویژه هستند. قطعی یا کندی این سرویس‌ها می‌تواند کل سامانه اطلاعات مکانی را مختل کند. چالش‌های عمده در پایش سرویس‌های مکانی عبارتند از:

- حجم بالای داده‌ها و نقشه‌ها

[۲۵] Alert

- نیاز به پاسخ سریع در درخواست‌های وب (WMS, WFS, WCS)
- اهمیت دقت و صحت داده‌ها
- تعداد بالای درخواست‌ها از جانب کاربران

یکی از ابزارهای تخصصی برای پایش سرویس‌های مکانی، GeoHealthCheck است. این ابزار می‌تواند به‌طور خودکار درخواست‌هایی به سرویس‌های OGC[۲۶] بفرستد و پاسخ آن‌ها را بررسی کند. اگر سرویس از دسترس خارج باشد یا پاسخ دیر بدهد، بلافاصله گزارش می‌دهد. برای مثال‌های پایش سرویس‌های مکانی می‌توان به موارد زیر اشاره نمود:

- بررسی سرویس WMS در ژئوسرور و اطمینان از صحت پاسخ نقشه‌ها
- بررسی سرویس WFS برای اطمینان از دسترسی به داده‌های برداری
- مانیتورینگ زمان پاسخ‌دهی سرویس‌ها برای جلوگیری از کندی در سیستم‌های بزرگ

۲-۳- ابزارهای رایج پایش

همانگونه که توضیح داده شد، پایش یکی از نیازهای اساسی در مدیریت سرورها و سرویس‌هاست. در سال‌های اخیر، ابزارهای متعددی برای پایش توسعه یافته‌اند؛ که در ادامه به بررسی رایج‌ترین آنها می‌پردازیم.

۲-۳-۱- آشنایی با Prometheus

یکی از برجسته‌ترین و پرکاربردترین ابزار پایش سرورها و سرویس‌های عمومی، Prometheus است. این ابزار که ابتدا توسط شرکت SoundCloud توسعه یافت و سپس به‌عنوان پروژه‌ای متن‌باز منتشر شد، امروزه به یکی از استانداردهای صنعتی برای پایش در محیط‌های مدرن، به‌ویژه زیرساخت‌هایی مانند سامانه‌های ابری[۲۷]، Docker[۲۸] و Kubernetes[۲۹] تبدیل شده است.

Prometheus نه‌تنها یک ابزار پایش ساده، بلکه یک سیستم کامل جمع‌آوری و تحلیل داده‌های عملکردی[۳۰] است. به‌عبارت دیگر، Prometheus داده‌های عملکردی را از سرویس‌ها جمع‌آوری می‌کند، آن‌ها را در یک پایگاه داده‌ی زمانی[۳۱] ذخیره می‌نماید، و امکان اجرای پرس‌وجوهای تحلیلی روی این داده‌ها را فراهم می‌کند. این سیستم پایش و هشداردهی متن‌باز است و متریک‌های

---

[۲۶] Open Geospatial Consortium

[۲۷] Cloud

[۲۸] بستری برای اجرای برنامه‌ها در محیط‌های جداگانه و قابل‌انتقال به نام Container

[۲۹] سامانه‌ای برای مدیریت، اجرا و مقیاس‌دهی Containerها در محیط‌های بزرگ و چندسروری

[۳۰] Metrics

[۳۱] Time-Series Database

جمع‌آوری شده معمولاً داده‌های کمی و قابل اندازه‌گیری هستند؛ مانند مصرف پردازنده، استفاده از حافظه، تعداد درخواست‌های وب، زمان پاسخ‌دهی سرویس‌ها، و بسیاری شاخص‌های دیگر.

Prometheus برخلاف ابزارهای قدیمی‌تر پایش، بر پایه‌ی مدل پایگاه داده‌ی سری زمانی کار می‌کند. در این مدل، داده‌ها همراه با برچسب‌های زمانی ذخیره می‌شوند و این امکان را می‌دهند که تغییرات آن‌ها در طول زمان تحلیل و بررسی شود. این ویژگی برای شناسایی روندها، الگوها و پیش‌بینی مشکلات آینده بسیار ارزشمند است.

ویژگی مهم دیگر Prometheus مدل جمع‌آوری داده‌ها[۳۲] است. به‌جای اینکه سرویس‌ها داده‌های خود را به سیستم پایش ارسال کنند، Prometheus به‌طور فعال به سرویس‌ها مراجعه کرده و داده‌ها را جمع‌آوری می‌کند. این شیوه باعث سادگی، انعطاف‌پذیری و مقیاس‌پذیری بالای سیستم می‌شود.

Prometheus **کاربردهای** بسیار گسترده‌ای دارد و به همین دلیل در محیط‌های متنوعی به کار گرفته می‌شود:

۱. پایش منابع سیستمی

یکی از اصلی‌ترین کارکردهای Prometheus جمع‌آوری متریک‌های پایه‌ای سرورهای لینوکسی یا ویندوزی است. این متریک‌ها شامل داده‌هایی مانند مصرف پردازنده، حافظه، فضای دیسک، و پهنای باند شبکه هستند. چنین متریک‌هایی پایه‌ی اصلی سلامت سرورها را تشکیل می‌دهند.

۲. پایش سرویس‌های نرم‌افزاری

Prometheus می‌تواند وضعیت سرویس‌های متعددی مانند وب‌سرورها، پایگاه‌های داده، یا سرویس‌های تخصصی را پایش کند. بسیاری از نرم‌افزارها، به‌ویژه آن‌هایی که در محیط‌های ابری[۳۳] و محفظه‌ای[۳۴] اجرا می‌شوند، خروجی‌های مخصوص[۳۵] برای Prometheus دارند تا داده‌های خود را در اختیار آن قرار دهند.

۳. پایش محیط‌های ابری و محفظه‌ای

در این محیط‌ها سرویس‌ها پویا هستند و به‌طور مداوم ایجاد یا حذف می‌شوند. Prometheus می‌تواند این تغییرات را به‌طور خودکار شناسایی و سرویس‌های جدید را بدون نیاز به پیکربندی دستی پایش کند؛ در واقع با تکیه بر مکانیزم‌های کشف سرویس، خود را با ایجاد و حذف مداوم سرویس‌ها تطبیق می‌دهد و پایش پیوسته را بدون مداخله‌ی انسانی ممکن می‌سازد.

---

[۳۲] Pull Model

[۳۳] Cloud

[۳۴] Container

[۳۵] Exporters

۴. هشداردهی (Alerting)

Prometheus دارای یک ماژول هشداردهی[۳۶] است که می‌تواند بر اساس قوانین از پیش‌تعریف‌شده، هشدارهای مختلفی تولید کند. برای مثال، اگر مصرف پردازنده یک سرور بیش از حد معمول باشد یا یک سرویس کلیدی از دسترس خارج شود، Prometheus هشدار صادر می‌کند و آن را از طریق ایمیل، پیام‌رسان‌ها یا سیستم‌های اعلان سازمانی ارسال می‌کند.

Prometheus دارای **مزایای** متعددی است که باعث شده به‌سرعت در جامعه‌ی فناوری اطلاعات محبوب شود:

- متن‌باز بودن و جامعه‌ی فعال: با توجه به اینکه این ابزار یک پروژه متن‌باز است و توسط بنیاد CNCF[۳۷] پشتیبانی می‌شود که توسعه‌دهندگان زیادی در سراسر دنیا روی آن کار می‌کنند.
- مدل داده‌های سری زمانی: ذخیره‌سازی داده‌ها به‌صورت سری زمانی امکان تحلیل دقیق روندها و شناسایی مشکلات پنهان را فراهم می‌کند.
- مقیاس‌پذیری بالا: به‌خاطر مدل جمع‌آوری داده‌ها و ساختار سبک، Prometheus می‌تواند در مقیاس بزرگ با هزاران سرویس به‌کار گرفته شود.
- سادگی در راه‌اندازی و استفاده: در مقایسه با ابزارهای سنتی مانیتورینگ، Prometheus بسیار ساده‌تر نصب و پیکربندی می‌شود.
- انعطاف‌پذیری بالا: با استفاده از زبان پرس‌وجوی اختصاصی خود (PromQL[۳۸])، کاربران می‌توانند گزارش‌ها و تحلیل‌های بسیار پیچیده تولید کنند.
- یکپارچگی با ابزارهای دیگر: Prometheus معمولاً همراه با Grafana استفاده می‌شود تا داده‌ها در قالب داشبوردهای گرافیکی نمایش داده شوند.

با وجود تمام مزایا، Prometheus بدون **محدودیت** نیست:

- ذخیره‌سازی بلندمدت محدود: پایگاه داده‌ی داخلی Prometheus برای ذخیره‌سازی کوتاه‌مدت بهینه شده است. برای نگهداری داده‌ها در بازه‌های زمانی طولانی، لازم است از سامانه‌های کمکی مخصوص ذخیره‌سازی بلندمدت استفاده شود.
- مقیاس‌پذیری افقی محدود: Prometheus به‌صورت پیش‌فرض به‌عنوان یک سرور منفرد طراحی شده است و پشتیبانی از خوشه‌بندی[۳۹] محدود دارد.

---

[۳۶] Alertmanager

[۳۷] Cloud Native Computing Foundation: بنیادی بین‌المللی و زیرمجموعه‌ی Linux Foundation است که از پروژه‌های متن‌باز مرتبط با زیرساخت‌های ابری، Container، Kubernetes، پایش و سامانه‌های توزیع‌شده پشتیبانی می‌کند.

[۳۸] Prometheus Query Language

[۳۹] Clustering: به‌کارگیری چند سرور به‌صورت هماهنگ به‌جای یک سرور منفرد

- مصرف منابع در محیط‌های بسیار بزرگ: در محیط‌هایی با ده‌ها هزار سرویس و میلیون‌ها متریک، مدیریت منابع می‌تواند چالش‌برانگیز باشد.
- تمرکز بر متریک‌ها، نه لاگ‌ها یا ردیابی‌ها: Prometheus عمدتاً برای جمع‌آوری متریک‌ها ساخته شده و به‌طور مستقیم برای بررسی لاگ‌ها[۴۰] (جزئیات متنی رویدادها) یا ردیابی درخواست‌ها[۴۱] (دنبال‌کردن مسیر کامل یک درخواست در میان چند سرویس) مناسب نیست.

همانطور که توضیح داده شد، Prometheus مانند یک دستگاه اندازه‌گیری دائمی برای سامانه‌ها عمل می‌کند. هر چیزی که قابل اندازه‌گیری باشد، از بار پردازنده گرفته تا تعداد درخواست‌های یک API[۴۲]، می‌تواند به‌وسیله‌ی Prometheus ثبت و تحلیل شود. **اهمیت** این کار در مدیریت سامانه‌های امروزی حیاتی است؛ در محیط‌های پویا مانند محیط‌های ابری، سرویس‌ها دائماً در حال تغییر هستند. بدون پایش هوشمند، مدیریت این تغییرات تقریباً غیرممکن است. در سامانه‌های مهم سازمانی، حتی چند ثانیه قطعی یا کندی می‌تواند خسارت مالی یا اعتباری به همراه داشته باشد. Prometheus با هشداردهی سریع مانع این اتفاق می‌شود. داده‌های سری زمانی که Prometheus ذخیره می‌کند، برای تحلیل‌های پیشگیرانه ارزشمندند. مدیران می‌توانند روندها را ببینند و قبل از وقوع بحران اقدام کنند. به همین دلایل است که Prometheus به یکی از ابزارهای اصلی پایش در اکوسیستم‌های بومی رایانش ابری[۴۳] تبدیل شده است و آشنایی با آن برای هر مدیر سیستم، مهندس عملیات توسعه[۴۴] یا توسعه‌دهنده‌ای که در حوزه‌ی سرویس‌های آنلاین و ابری فعالیت دارد، ضروری است.

۲-۳-۲- آشنایی با Grafana

در مباحث مدیریت زیرساخت‌ها و سرویس‌های نرم‌افزاری، داشتن اطلاعات دقیق و به‌موقع از وضعیت سامانه‌ها تنها نیمی از راه است. نیمی دیگر توانایی نمایش و تفسیر این داده‌ها به شکلی قابل فهم و کارآمد است. داده‌های خام مانند اعداد مربوط به مصرف پردازنده یا ترافیک شبکه اگرچه ارزشمندند، اما بدون نمایش بصری، درک آن‌ها برای تصمیم‌گیری سریع دشوار است.

اینجاست که ابزارهایی مانند Grafana اهمیت پیدا می‌کنند. Grafana نه یک ابزار جمع‌آوری داده، بلکه یک بستر قدرتمند برای بصری‌سازی داده‌ها و ایجاد داشبوردهای پویا است. این نرم‌افزار متن‌باز به مدیران سیستم، مهندسان DevOps و حتی مدیران سازمان‌ها

---

[۴۰] Log Monitoring

[۴۱] Tracing

[۴۲] Application Programming Interface: رابط برنامه‌نویسی کاربردی

[۴۳] Cloud Native

[۴۴] DevOps: مخفف نام دو تیم مستقل، توسعه نرم‌افزار Development و عملیات فناوری اطلاعات Operations است و به مجموعه‌ای از اقداماتی گفته می‌شود که بر ارتباط و همکاری تیم‌های توسعه نرم‌افزار تمرکز دارند تا فرآیند توسعه نرم‌افزار بهتر و سریع‌تر انجام شود.

این امکان را می‌دهد که وضعیت سیستم‌های پیچیده را در قالب نمودارها، جداول و گراف‌ها مشاهده کنند و با یک نگاه، سلامت و عملکرد زیرساخت را ارزیابی نمایند.

Grafana یک نرم‌افزار متن‌باز برای بصری‌سازی[۴۵] و تحلیل[۴۶] داده‌هاست که در اصل برای نمایش متریک‌های جمع‌آوری‌شده توسط ابزارهایی مانند Prometheus طراحی شده است. این ابزار داده‌ها را از منابع مختلف[۴۷] دریافت کرده و آن‌ها را در قالب داشبوردهای تعاملی و زیبا نمایش می‌دهد.

یکی از ویژگی‌های کلیدی Grafana این است که محدود به یک منبع داده خاص نیست. برخلاف ابزارهایی که فقط با پایگاه داده یا سیستم مانیتورینگ خاصی کار می‌کنند، Grafana می‌تواند داده‌ها را از طیف وسیعی از منابع بخواند: از Prometheus و InfluxDB گرفته تا Elasticsearch، MySQL، PostgreSQL و حتی فایل‌های ساده‌ی CSV. همین تنوع باعث شده که Grafana به یک ابزار پرکاربرد در حوزه‌های مختلف تبدیل شود.

کاربرد اصلی Grafana در نمایش و تحلیل بصری داده‌ها است. اهم توانمندی‌های این ابزار عبارتند از:

۱. ایجاد داشبوردهای پویا

Grafana به کاربران اجازه می‌دهد داشبوردهایی با نمودارهای مختلف طراحی کنند. این داشبوردها می‌توانند اطلاعات حیاتی مانند بار پردازنده، مصرف حافظه، زمان پاسخ سرویس‌ها، تعداد درخواست‌ها و وضعیت سلامت بستر سرویس‌های مکانی (مانند ژئوسرور) را نمایش دهند. داشبوردها پویا هستند و امکان فیلتر کردن و تغییر بازه‌های زمانی به‌صورت تعاملی وجود دارد.

۲. تحلیل روندها

با استفاده از داده‌های سری زمانی که از منابعی مانند Prometheus دریافت می‌شوند، Grafana می‌تواند روندها را به‌خوبی نمایش دهد. برای مثال، مدیر سیستم می‌تواند الگوی مصرف پردازنده در طول شبانه‌روز یا افزایش تدریجی بار ترافیک را مشاهده و از این اطلاعات برای پیش‌بینی آینده استفاده کند.

۳. پایش سرویس‌های حیاتی

در سازمان‌هایی که سرویس‌های آنلاین برای ارائه‌ی خدمات حیاتی دارند، داشتن یک نگاه سریع به وضعیت سیستم بسیار مهم است. Grafana با ارائه‌ی داشبوردهای بزرگ و واضح، امکان پایش سریع سرویس‌های کلیدی را فراهم می‌کند. به همین دلیل است که در بسیاری از اتاق‌های عملیات، نمایشگرهای بزرگی نصب می‌شود که داشبوردهای Grafana را نشان می‌دهند.

---

[۴۵] Visualization

[۴۶] Analytics

[۴۷] Data Sources

۴. کاربرد فراتر از IT

اگرچه Grafana در ابتدا برای پایش سامانه‌های IT توسعه یافته، اما امروزه در حوزه‌های دیگری هم استفاده می‌شود. برای مثال، در صنایع تولیدی برای داده‌های مربوط به خطوط تولید؛ در حوزه‌ی سلامت برای داده‌های مرتبط با بیماران؛ یا در تجارت برای شاخص‌های مالی، همه می‌توانند توسط Grafana بصری‌سازی شوند.

دلایل متعددی وجود دارد که Grafana را به انتخاب اول بسیاری از سازمان‌ها برای بصری‌سازی تبدیل کرده است. از جمله این **مزایا** میتوان به موارد ذیل اشاره نمود:

- متن‌باز بودن و جامعه‌ی فعال: Grafana متن‌باز است و توسعه‌دهندگان زیادی به‌طور مداوم آن را بهبود می‌دهند.
- پشتیبانی از منابع داده‌ی متنوع: Grafana می‌تواند داده‌ها را از بیش از ۳۰ منبع مختلف بخواند.
- داشبوردهای تعاملی و قابل سفارشی‌سازی: کاربران می‌توانند داشبوردهای دلخواه خود را طراحی کنند و آن‌ها را با دیگران به اشتراک بگذارند.
- بصری‌سازی پیشرفته: از نمودارهای ساده‌ی خطی تا گراف‌های پیچیده، همه در Grafana قابل طراحی‌اند.
- هشداردهی: Grafana امکان تعریف هشدارهای هوشمند دارد که می‌توانند بر اساس داده‌های دریافتی، اعلان‌هایی از طریق ایمیل یا پیام‌رسان‌ها ارسال کنند.

با وجود همه‌ی مزایا، Grafana نیز **محدودیت‌های** خاص خود را دارد:

- عدم جمع‌آوری داده‌ها: Grafana به‌تنهایی قادر به جمع‌آوری داده‌ها نیست و حتماً نیاز به یک منبع داده مانند Prometheus یا پایگاه داده دارد.
- پیچیدگی در داشبوردهای بزرگ: اگر تعداد نمودارها و داده‌ها بسیار زیاد باشد، طراحی داشبوردهای کارآمد دشوار می‌شود.
- وابستگی به کیفیت داده‌ها: Grafana فقط داده‌ها را نمایش می‌دهد. اگر داده‌های ورودی ناقص یا اشتباه باشند، خروجی گمراه‌کننده خواهد بود.
- مصرف منابع در نمایش لحظه‌ای: در محیط‌هایی که داده‌ها در مقیاس بزرگ و با سرعت بالا تغییر می‌کنند، نمایش لحظه‌ای می‌تواند فشار زیادی به منابع سیستم وارد کند.

در نتیجه می‌توان گفت Grafana در اصل یک پنجره‌ی شفاف برای مشاهده‌ی وضعیت سامانه‌هاست. در حالی که ابزارهایی مانند Prometheus داده‌ها را جمع‌آوری و ذخیره می‌کنند، Grafana آن داده‌ها را قابل فهم و قابل استفاده می‌سازد.

اهمیت Grafana از اینجا ناشی می‌شود که تصمیم‌گیری سریع و درست بدون نمایش مناسب تقریباً غیرممکن است. مدیران و تیم‌های فنی با نگاه به یک داشبورد Grafana می‌توانند فوراً بفهمند که کدام سرویس مشکل دارد، بار کدام سرور غیرعادی است، یا روند مصرف منابع در چه جهتی حرکت می‌کند.

به همین دلیل، Grafana نه‌تنها یک ابزار فنی برای تیم‌های فناوری اطلاعات، بلکه یک ابزار مدیریتی برای سطح‌های بالاتر سازمان نیز محسوب می‌شود؛ لذا آشنایی با این ابزار برای هر کسی که در حوزه‌ی مدیریت سیستم‌ها، توسعه یا حتی تحلیل داده فعالیت دارد، ضروری است.

۲-۳-۳- آشنایی با GeoHealthCheck

در دنیای امروز، سرویس‌های مکانی و نقشه‌ای بخش جدایی‌ناپذیر بسیاری از سامانه‌ها و سازمان‌ها هستند. چه در حوزه‌ی مدیریت شهری، چه در برنامه‌ریزی محیطی، چه در کاربردهای تجاری و حتی در حوزه‌هایی چون سلامت، حمل‌ونقل و زیرساخت‌ها، داده‌های مکانی نقش کلیدی ایفا می‌کنند. بسیاری از این داده‌ها از طریق استانداردهای OGC[۴۸] در دسترس قرار می‌گیرند؛ استانداردهایی مانند WMS[۴۹]، WFS[۵۰] و WCS[۵۱].

با افزایش اهمیت این سرویس‌ها، یک پرسش بنیادین پیش می‌آید: چگونه مطمئن شویم این سرویس‌ها همیشه در دسترس‌اند و درست کار می‌کنند؟ به این سبب است که پایش وارد عمل می‌شود. پایش سرویس‌های مکانی به مدیران کمک می‌کند وضعیت سرویس‌ها را به طور مداوم بررسی کنند، مشکلات احتمالی را زودتر کشف کنند و از بروز قطعی یا افت کیفیت سرویس جلوگیری نمایند.

یکی از ابزارهای تخصصی که برای این کار توسعه یافته است، GeoHealthCheck (GHC) نام دارد. این ابزار یک سامانه‌ی پایش برای سرویس‌های OGC است که با رویکردی تخصصی و دقیق طراحی شده و یک نرم‌افزار متن‌باز است. این ابزار می‌تواند به شکل خودکار درخواست‌هایی به سرویس‌های OGC ارسال کند و کیفیت پاسخ آن‌ها را بسنجد. اگر سرویس در دسترس نباشد، پاسخ نادرست بدهد یا کندتر از حد انتظار عمل کند، GeoHealthCheck آن را ثبت کرده و هشدار می‌دهد.

برخلاف ابزارهای عمومی پایش که تنها بررسی می‌کنند یک پورت یا سرویس روشن است یا خیر، GeoHealthCheck دقیقاً متناسب با استانداردهای مکانی ساخته شده است. برای مثال می‌تواند بررسی کند که:

- سرویس WMS به درخواست استاندارد دریافت نقشه پاسخ می‌دهد و پاسخ آن دچار خطای سرویس یا قالب نامعتبر نیست.
- سرویس WFS به درخواست دریافت داده پاسخ معتبر بازمی‌گرداند و پاسخ از نظر ساختار قابل پردازش است.
- سرویس WCS درخواست دریافت داده شبکه‌ای و پوششی[۵۲] را بدون خطای پردازشی اجرا می‌کند.

---

[۴۸] Open Geospatial Consortium

[۴۹] Web Map Service

[۵۰] Web Feature Service

[۵۱] Web Coverage Service

[۵۲] Coverage

به این ترتیب GeoHealthCheck نه تنها در دسترس بودن سرویس را بررسی می‌کند، بلکه صحت عملکرد آن را نیز تضمین می‌نماید. لذا **کارکرد اصلی** GeoHealthCheck را می‌توان در سه محور اصلی خلاصه کرد:

۱. بررسی در دسترس بودن سرویس‌ها

GHC به صورت دوره‌ای (مثلاً هر چند دقیقه یکبار) درخواست‌هایی به سرویس‌های مکانی می‌فرستد. اگر سرویس پاسخگو نباشد، بلافاصله آن را به عنوان خطا ثبت می‌کند. این ویژگی کمک می‌کند مدیران از قطعی سرویس‌ها در همان لحظه مطلع شوند.

۲. ارزیابی کیفیت و صحت پاسخ‌ها

صرفاً در دسترس بودن سرویس کافی نیست. ممکن است سرویس پاسخی با خطای فنی ارائه دهد. GeoHealthCheck با اجرای درخواست‌های عملیاتی از پیش تعریف‌شده، بررسی می‌کند که سرویس بتواند پارامترهایی مانند نام لایه، سیستم مختصات و محدوده‌ی مکانی را از نظر فنی پردازش کرده و پاسخ معتبر تولید کند.

۳. تحلیل کارایی و زمان پاسخ‌دهی

یکی از معیارهای مهم در سرویس‌های مکانی، زمان پاسخ‌دهی است. اگر یک سرویس WMS برای نمایش یک نقشه ساده بیش از چند ثانیه زمان ببرد، کاربران تجربه‌ی خوبی نخواهند داشت. GeoHealthCheck زمان پاسخ سرویس‌ها را ثبت کرده و روند تغییرات آن را نشان می‌دهد. این داده‌ها برای بهینه‌سازی زیرساخت و منابع بسیار مفیدند و به‌طور خاص بر جنبه‌ی کارایی سرویس تمرکز داشته و مستقل از در دسترس‌بودن یا اعتبار فنی پاسخ هستند.

GHC به عنوان یک ابزار تخصصی پایش، دارای **مزایا**ی متعددی است که آن را از دیگر ابزارها متمایز می‌کند:

- تخصصی بودن برای OGC: برخلاف مانیتورینگ‌های عمومی، GHC دقیقاً برای استانداردهای OGC طراحی شده و توانایی تست کردن پارامترها و ویژگی‌های خاص آن‌ها را دارد.
- متن‌باز بودن: این ابزار آزاد و رایگان است و جامعه‌ی کاربری فعالی دارد. هر کسی می‌تواند آن را توسعه دهد، تغییر دهد یا متناسب با نیاز خود سفارشی‌سازی کند.
- نمایش گرافیکی وضعیت: GHC یک رابط وب ساده و کاربرپسند ارائه می‌دهد که وضعیت سرویس‌ها، زمان‌های پاسخ‌دهی و تاریخچه‌ی خطاها را به‌صورت گرافیکی نمایش می‌دهد.
- هشداردهی سریع: در صورت بروز خطا یا قطعی، GHC با استفاده از سازوکار اعلان قابل پیکربندی، رویداد خطا را از طریق کانال‌های تعریف‌شده مانند ایمیل، به مدیر سیستم اطلاع می‌دهد تا مشکل در کوتاه‌ترین زمان رفع شود.
- ذخیره‌سازی تاریخچه: تمام نتایج تست‌ها ذخیره می‌شوند و امکان تحلیل بلندمدت برای شناسایی الگوهای خطا یا افت کیفیت وجود دارد.

البته مانند هر ابزار دیگری، GeoHealthCheck نیز **محدودیت‌ها**یی دارد:

- تمرکز محدود روی OGC: این ابزار بیشتر برای سرویس‌های OGC کاربرد دارد و برای پایش سرویس‌های عمومی یا غیرمکانی مناسب نیست.
- پیچیدگی در پیکربندی پیشرفته: در حالی که استفاده‌ی ساده از آن آسان است، اما تنظیمات پیشرفته برای سرویس‌های پیچیده می‌تواند زمان‌بر باشد.
- نیاز به منابع برای حجم بالا: در سازمان‌هایی با ده‌ها یا صدها سرویس، بار تست‌ها می‌تواند زیاد شود و نیاز به منابع سخت‌افزاری بیشتری داشته باشد.

با این حال، این محدودیت‌ها در مقایسه با مزایای آن چندان جدی نیستند و معمولاً با روش‌های مدیریتی قابل رفع‌اند.

GeoHealthCheck نقشی مشابه یک پزشک برای سرویس‌های مکانی دارد. همان‌طور که یک پزشک سلامت بدن را مرتب بررسی می‌کند، GHC سلامت سرویس‌های OGC را به‌طور پیوسته بررسی می‌نماید. **اهمیت** این کار زمانی آشکار می‌شود که بدانیم بسیاری از سرویس‌های مکانی در پروژه‌های حیاتی مانند سامانه‌های مدیریت بحران (زلزله، سیل، آتش‌سوزی)، سامانه‌های حمل‌ونقل و مسیریابی، سیستم‌های سلامت و پایش محیطی، سامانه‌های برنامه‌ریزی شهری و کشاورزی و سایر استفاده می‌شوند. در چنین مواردی حتی چند دقیقه قطعی یا خطا می‌تواند خسارت‌های جدی به بار آورد. GeoHealthCheck کمک می‌کند این سرویس‌ها همیشه پایدار، سریع و قابل اعتماد باقی بمانند.

فصل سوم

راه‌اندازی سرور پایش‌شونده

۳-۱- راه‌اندازی سرویس‌های مکانی

سرورِ «پایش‌شونده» در این‌جا به سروری اطلاق می‌شود که هم میزبان سرویس‌های مکانی و بستر ارائه آنها (مانند ژئوسرور[۵۳]) است و هم تحت نظارت مستمر ابزارهای مانیتورینگ (Prometheus/Grafana/GeoHealthCheck) قرار دارد. چنین سروری معمولاً در نقش ارائه‌دهنده‌ی داده‌های مکانی کار می‌کند و بنابراین نیازمند پایداری، پاسخ‌دهی سریع و قابلیت بازیابی است. هنگام طراحی و راه‌اندازی، باید از همان ابتدا به نیازهای عملکردی (پردازنده، حافظه، دیسک، شبکه)، امنیت (فایروال، دسترسی کاربری، ایمن‌سازی ارتباطات[۵۴]) و قابلیت مانیتورینگ (لاگ‌گذاری، متریک‌ها، سلامت سرویس) توجه شود؛ زیرا داده‌های مکانی معمولاً حجم بالا و الگوهای بار نامتوازن[۵۵] دارند و تاخیر یا خرابی در سرویس می‌تواند بطور جدی بر تجربه‌ی کاربران یا فرایندهای بالادستی اثر بگذارد.

انتخاب تامکت[۵۶] و ژئوسرور برای اجرای سامانه و ارائه‌ی سرویس‌های مکانی دلایل فنی و عملی روشنی دارد. ژئوسرور یک سرور نقشه‌ی متن‌باز است که استانداردهای OGC را پیاده‌سازی می‌کند؛ یعنی می‌تواند به‌طور مستقیم سرویس‌های WMS/WFS/WCS ارائه دهد، با منابع متداول مکانی (مانند PostGIS و داده‌های رستری نظیر GeoTIFF) کار کند، و از قابلیت‌هایی مانند SLD برای استایلینگ و سبک‌دهی لایه‌ها و GeoWebCache برای ذخیره‌سازی موقت کاشی[۵۷]های نقشه پشتیبانی می‌کند. از این‌رو برای سازمان‌هایی که می‌خواهند سرویس‌های استاندارد در اکوسیستم اطلاعات مکانی ارائه دهند، ژئوسرور گزینه‌ای شناخته‌شده و بالغ است. تامکت نیز یک محیط اجرایی جاوا است که امکان استقرار و اجرای برنامه‌های وب بسته‌بندی‌شده در قالب فایل‌های WAR را فراهم می‌کند؛ ژئوسرور به‌صورت یک وب‌اپلیکیشن جاوایی منتشر می‌شود و تامکت محیطی سبک، پایدار و آشنا برای اجرای آن فراهم می‌کند. ترکیب این دو یعنی استفاده از یک زیرساخت جاوایی استاندارد که مدیریت، به‌روزرسانی و مقیاس‌پذیری آن نسبتاً ساده است و جامعه‌ی بزرگی برای پشتیبانی و افزونه‌ها وجود دارد.[۵۸]

راه‌اندازی سرویس‌های مکانی با استفاده از تامکت و ژئوسرور در سیستم‌عامل لینوکس یک فرآیند چند مرحله‌ای است که شامل نصب و پیکربندی و آماده‌سازی اجزای لازم برای اجرای سامانه می‌باشد. در مورد پیش‌نیازها و شرایط عملیاتی که قبل از راه‌اندازی باید در نظر گرفته شود، چند نکته‌ی کلیدی وجود دارد: سرور باید نسخه‌ی سازگار محیط اجرایی جاوا[۵۹] را داشته باشد، حافظه‌ی کافی برای

---

[۵۳] GeoServer

[۵۴] TLS (Transport Layer Security): پروتکلی برای رمزنگاری و تضمین امنیت ارتباطات

[۵۵] میزان و نوع درخواست‌های وارده به سامانه در طول زمان یا بین کاربران مختلف یکنواخت نبوده و به‌صورت مقطعی یا ناپایدار تغییر می‌کند.

[۵۶] Tomcat

[۵۷] Tile

[۵۸] جایگزین‌هایی مانند Jetty هم وجود دارند، اما تامکت به‌خاطر سابقه و ابزارهای مدیریتی معمولاً انتخاب متداول‌تری است.

[۵۹] JDK/JRE

اجرای ماشین مجازی جاوا[۶۰] و پردازش درخواست‌های نقشه فراهم باشد، پردازنده با هسته‌های کافی برای پاسخ‌دهی هم‌زمان و دیسک با کارآیی مناسب برای نگهداری و خواندن داده‌های حجیم و رستری[۶۱] را داشته و همچنین فضای کافی برای ذخیره‌سازی موقت[۶۲] سامانه لازم است. مسیر ذخیره‌ی داده‌ی ژئوسرور[۶۳] باید مشخص و ترجیحا مجزا و دارای دسترسی خواندن و نوشتن برای کاربر اجرا کننده سرویس باشد و سیاست‌های پشتیبان‌گیری[۶۴] منظم برای آن تعریف شود.

از منظر شبکه و امنیت، لازم است پورت اجرای تامکت (مانند پورت پیش‌فرض ۸۰۸۰) مدیریت و در صورت نیاز با پروتکل ایمن‌سازی ارتباطات (از طریق HTTPS) امن شود؛ حساب‌های کاربری مرتبط با اجرای سرویس باید محدود بوده و مجوزهای دسترسی به‌درستی تنظیم شوند. برای سروری که «بدون اینترنت» فعالیت می‌کند، فرایند تهیه‌ی بسته‌های لازم (مانند JDK، تامکت، ژئوسرور و افزونه‌ها، درایورها و سایر) باید از قبل انجام شده و به صورت آفلاین منتقل و کنترل نسخه شوند. همچنین برای پایش بهتر، لازم است سازوکارهای لاگ‌گذاری فعال و قابل خواندن باشند و در صورت نیاز، خروجی‌هایی برای متریک‌ها فعال شود تا ابزارهای مانیتورینگ بتوانند داده‌های مربوط به سلامت و عملکرد را جمع‌آوری کنند.

این مستند به طور جامع مراحل انجام کار را بر روی سرورهای سازمانی با مثال‌هایی از داده‌های سازمان نقشه‌برداری کشور شرح می‌دهد.

مراحل انجام کار به‌صورت گام‌به‌گام در ادامه آورده شده‌اند:

مرحله ۱: درخواست دسترسی به سرور پایش‌شونده

به منظور دسترسی به مدیریت یک سرور، ابتدا باید امکان دسترسی توسط واحد فناوری اطلاعات برای کاربر ایجاد شود.

مرحله ۲: ورود به سرور پایش‌شونده

پس از اخذ دسترسی، گام بعدی ورود به محیط اسکریپت‌نویسی لینوکسی در سرور پایش‌شونده است که از مسیر مشخص‌شده توسط واحد فناوری اطلاعات طی می‌گردد.

۳-۱-۱- نصب Apache Tomcat

تامکت (Apache Tomcat) در واقع یک محیط اجرایی برای برنامه‌های وب جاوا است که امکان استقرار و اجرای برنامه‌های تحت وب نوشته‌شده با جاوا را فراهم می‌کند. از آن‌جا که ژئوسرور به‌صورت یک برنامه‌ی وب جاوایی (در قالب فایل WAR) منتشر

---

[۶۰] JVM (Java virtual machine)

[۶۱] Raster Data

[۶۲] Cache

[۶۳] data directory

[۶۴] Backup

می‌شود، نیاز دارد در یک محیط اجرایی سازگار با جاوا مستقر شود. به همین دلیل پیش از نصب و راه‌اندازی ژئوسرور، باید تامکت روی سرور نصب و پیکربندی شود تا ژئوسرور در آن مستقر[۶۵] و اجرا گردد.

مرحله ۳: بروزرسانی فهرست بسته‌های لینوکس

پیش از نصب هرگونه بسته نرم‌افزاری، ابتدا باید فهرست بسته‌ها[۶۶]ی سیستم عامل را بروزرسانی کرد. این فهرست‌ها شامل اطلاعات محلی مربوط به بسته‌های نرم‌افزاری و نسخه‌ها و وابستگی‌های آن‌ها است که از مخازن سیستم‌عامل دریافت می‌شود. به‌روزرسانی این فهرست پیش از نصب پکیج‌ها ضروری است، زیرا در غیر این صورت سیستم ممکن است به نسخه‌های قدیمی ارجاع دهد یا هنگام شناسایی و نصب وابستگی‌ها با خطا مواجه شود. به این منظور از دستور زیر استفاده می‌شود.

```
sudo apt update                    # بروزرسانی لیست بسته‌ها
[sudo] password for user:          پس از اجرای دستور فوق، رمز کاربر سوال می‌شود که هنگام ورود آن نیز کاراکترها
                                   نمایش داده نمی‌شوند ولی در پایان با دکمه Enter اعمال می‌گردد.
```

البته جهت اجرای موفق دستور فوق، لازم است سرور از طریق اینترنت به مخازن سیستم عامل دسترسی داشته باشد؛ در غیر اینصورت پیغام خطای عدم موفقیت بروزرسانی‌ها نمایش داده خواهد شد. با لحاظ فرضیه آپدیت بودن فهرست بسته‌ها، به گام بعدی می‌رویم.

مرحله ۴: اطمینان از نصب جاوا[۶۷]

از آنجا که تامکت خودش یک برنامه‌ی جاوایی است و برای اجرا به جاوا نیاز دارد، پیش‌نیاز نصب و راه‌اندازی آن، نصب جاوا روی سرور است. به بیان ساده، جاوا محیط اجرایی پایه را فراهم می‌کند تا تامکت بتواند اجرا شده و در ادامه ژئوسرور را میزبانی کند.

برای اطمینان از وجود یا عدم وجود جاوا روی سرور از دستور زیر استفاده می‌شود:

```
java --version                     # مشاهده ورژن جاوا در صورت نصب بودن یا پیغام خطا در صورت نصب نبودن
```

```
mehranfar_m@gsb-linux:~$ java --version
openjdk 11.0.26 2025-01-21
OpenJDK Runtime Environment (build 11.0.26+4-post-Ubuntu-1ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.26+4-post-Ubuntu-1ubuntu122.04, mixed mode, sharing)
```

شکل ۳-۱ – نمونه پیغام خروجی حاصل از دستور ورژن‌یابی در صورت نصب بودن جاوا

---

[۶۵] deploy

[۶۶] Package index

[۶۷] Java

```
esmaeilsaadatzadeh@IRANSDI:~$ java -version
Command 'java' not found, but can be installed with:
sudo apt install default-jre              # version 2:1.11-72build2, or
sudo apt install openjdk-11-jre-headless  # version 11.0.24+8-1ubuntu3~22.04
sudo apt install openjdk-17-jre-headless  # version 17.0.12+7-1ubuntu2~22.04
sudo apt install openjdk-18-jre-headless  # version 18.0.2+9-2~22.04
sudo apt install openjdk-19-jre-headless  # version 19.0.2+7-0ubuntu3~22.04
sudo apt install openjdk-21-jre-headless  # version 21.0.4+7-1ubuntu2~22.04
sudo apt install openjdk-8-jre-headless   # version 8u422-b05-1~22.04
```

شکل ۳-۲ - پیغام خروجی حاصل از دستور ورژن‌یابی در صورت نصب نبودن جاوا

اگر جاوا نصب بوده باشد، برای نصب تامکت (مطابق با ورژن جاوای موجود) ورود به مرحله ۵: امکان‌پذیر خواهد بود.

در صورت نصب نبودن جاوا، با دستور زیر می‌توان پکیج مربوطه را دانلود و نصب نمود. البته نیاز است سرور به اینترنت دسترسی داشته باشد؛ در غیر اینصورت بایستی پکیج را در سیستمی که دسترسی به اینترنت دارد، از سایت رسمی جاوا دانلود کرده و به سرور منتقل نمود و سپس بصورت آفلاین نصب کرد.

```
sudo apt install openjdk-11-jdk        # نصب آنلاین جاوا
```

مرحله ۵: دانلود تامکت

با توجه به اینکه اغلب سرورهای سازمانی، محدودیت دسترسی به اینترنت آزاد دارند، بایستی بسته نرم‌افزاری تامکت در ورژنی متناسب با ورژن جاوا را، در سیستمی که دسترسی به اینترنت دارد (رایانه شخصی) دانلود نمود و سپس فایل دانلود شده به سرور و داخل پوشه مدنظر (مانند پوشه Downloads) منتقل گردد. برای دانلود به آدرس زیر می‌توان مراجعه نمود:

https://downloads.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.tar.gz

مرحله ۶: انتقال فایل دانلود شده تامکت به سرور

فایل دانلود شده بایستی به‌صورت آفلاین به سرور هدف منتقل گردد. روش انجام این کار، با توجه به زیرساخت موجود و سیاست‌های امنیتی هر سازمان، توسط واحد فناوری اطلاعات آن تعیین می‌شود.

مرحله ۷: نصب تامکت

پس از اینکه فایل در مسیر مدنظر روی سرور (پوشه Downloads) قرار گرفت، وارد پوشه مربوطه شده و با دستور زیر فایل دانلود شده از حالت فشرده خارج و محتوای آن استخراج می‌شود:

```
cd Downloads                                  # ورود به پوشه حاوی فایل دانلود شده تامکت
tar xvzf apache-tomcat-9.0.108.tar.gz         # خارج کردن از حالت فشرده و استخراج محتوا
```

مرحله ۸: مدیریت دسترسی تامکت

با وارد کردن دستور ls محتویات پوشه‌ای که تامکت در آن استخراج شد، قابل مشاهده می‌باشد. رنگ آبی پوشه تامکت نمایانگر دایرکتوری تامکت و رنگ قرمز نمایانگر فایل فشرده آن است (شکل ۳-۳).

```
mehranfar_m@gsb-linux:~/Downloads$ ls
apache-tomcat-9.0.108  apache-tomcat-9.0.108.tar.gz
mehranfar_m@gsb-linux:~/Downloads$ ls apache*
apache-tomcat-9.0.108.tar.gz

apache-tomcat-9.0.108:
bin  BUILDING.txt  conf  CONTRIBUTING.md  lib  LICENSE  logs  NOTICE  README.md  RELEASE-NOTES  RUNNING.txt  temp  webapps  work
```

شکل ۳-۳ – محتوی پوشه‌ی تامکت

برای اعطای دسترسی اجرا به دایرکتوری تامکت و سبز رنگ شدن آن دستورهای زیر وارد می‌شود؛ در ابتدا نیز برای راحتی در فراخوانی نام دایرکتوری، دایرکتوری تامکت تغییر نام می‌یابد:

```
mv  apache-tomcat-9.0.108 tomcat9          # تغییر نام پوشه apache-tomcat-9.0.108 به tomcat9
sudo chmod -R 755 ~/Downloads/tomcat9      # اعطای دسترسی کامل به مالک پوشه
sudo chown -R user:user ~/Downloads/tomcat9 # اعطای دسترسی به کاربر
                                           بایستی به جای user ها نام کاربری شخصی وارد شود
```

در این مرحله نصب تامکت پایان‌یافته تلقی می‌گردد که برای راه‌اندازی آن بایستی از داخل فولدر bin و از پوشه tomcat9 دستور sudo ./startup.sh وارد می‌شود تا تامکت شروع به فعالیت کند. ولی بهتر است راه‌اندازی تامکت پس از مستقر کردن[۶۸] ژئوسرور انجام پذیرد.

تذکر ۱:

اگر سرور هدف به اینترنت آزاد دسترسی داشته باشد، می‌توان به‌جای مرحله ۵: و مرحله ۶: با استفاده از دستور زیر، تامکت را مستقیما روی سرور دانلود کرد و ادامه راه را از مرحله ۷: طی کرد:

```
sudo wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.tar.gz
```

۳-۱-۲- نصب GeoServer

بعد از نصب و راه‌اندازی تامکت، ژئوسرور به‌عنوان یک برنامه‌ی وب (WAR) روی آن مستقر می‌شود و از طریق مرورگر و پورت مربوط به تامکت (پیش‌فرض، ۸۰۸۰) قابل دسترسی خواهد بود. ژئوسرور در این مرحله نقش اصلی خود یعنی ارائه‌ی سرویس‌های مکانی

[۶۸] Deploy

با استاندارد OGC را ایفا می‌کند. به این ترتیب سرور لینوکسی به یک بستر کامل برای انتشار و مدیریت داده‌های مکانی در شبکه تبدیل می‌شود.

مرحله ۹: دانلود ژئوسرور

در این گام باید دقت نمود که برای ژئوسرور، ورژن متناسب با نسخه جاوا و تامکت را تهیه کرده و سپس در پوشه webapps تامکت مستقر نمود. با توجه به اینکه اغلب سرورهای سازمانی دسترسی به اینترنت ندارد، بایستی از مرورگر رایانه شخصی اقدام به دانلود ژئوسرور نمود که نمونه مخزن دانلود از لینک زیر در دسترس است:

https://sourceforge.net/projects/geoserver/files/GeoServer/2.24.2/geoserver-2.24.2-war.zip/download

مرحله ۱۰: انتقال فایل دانلود شده‌ی ژئوسرور به تامکت

برای راه‌اندازی ژئوسرور، بایستی آن‌را داخل پوشه webapps از پوشه tomcat9 مستقر کرد. به‌این‌منظور با استفاده از پروتکل تعریف‌شده از سوی واحد فناوری اطلاعات فایل از رایانه شخصی به مسیر Downloads/tomcat9/webapps/ بارگذاری[۶۹] می‌شود.

مرحله ۱۱: خارج نمودن از حالت فشرده فایل ژئوسرور

با دستورات زیر از داخل پوشه webapps فایل دانلود شده‌ی ژئوسرور را از حالت فشرده خارج نموده که در نتیجه آن فایل geoserver.war داخل پوشه webapps استخراج می‌شود و با اولین اجرای تامکت، فایل WAR به‌صورت خودکار مستقر شده و پوشه‌ای با نام geoserver نیز ایجاد می‌شود.

```
cd tomcat9/webapps                  # ورود به پوشه حاوی فایل دانلود شده ژئوسرور
unzip geoserver-2.24.2-war.zip      # خارج کردن از حالت فشرده
```

مرحله ۱۲: تخصیص حافظه

برای راه‌اندازی سرویس‌های مکانی با تعداد زیاد و یا حجم بالای داده‌ها، نیاز است مقادیر پیش‌فرض حافظه‌ی تخصیص یافته به تامکت و ژئوسرور، متناسب با منابع سرور تنظیم گردد. بدین منظور:

```
echo 'export JAVA_OPTS="-Xms512m -Xmx2048m"' > ~/Downloads/tomcat9/bin/setenv.sh
chmod +x ~/Downloads/tomcat9/bin/setenv.sh
```

این دو دستور درکنار هم، اول یک فایل اسکریپت به نام setenv.sh داخل پوشه‌ی bin تامکت می‌سازند که متغیرهای محیطی جاوا را در آن قرار می‌دهد؛ در این متغیر مقدار Xms حداقل حافظه‌ی اولیه و مقدار Xmx حداکثر حافظه‌ی قابل استفاده توسط ماشین مجازی جاوا را مشخص می‌کند و سپس با chmod +x آن فایل را قابل اجرا می‌نمایند.

---

[۶۹] Upload

مرحله ۱۳: راه‌اندازی تامکت و ژئوسرور

در این مرحله وارد پوشه bin تامکت شده و با اجرای[۷۰] دستی تامکت، فرآیند استقرار و اجرای ژئوسرور اتمام می‌یابد. همچنین فایل shutdown.sh برای توقف[۷۱] تامکت نیز در همین مسیر bin می‌باشد که در مواقع نیاز می‌توان به آن مراجعه نمود و همانند دستور زیر عمل کرد.

```
cd ~/Downloads/tomcat9/bin                  # ورود به پوشه bin در تامکت
sudo ./startup.sh                           # راه‌اندازی تامکت که در صورت لزوم رمز کاربر وارد شود
```

مرحله ۱۴: اتوماتیک‌سازی راه‌اندازی تامکت و ژئوسرور

به‌منظور اینکه در هر بار راه‌اندازی مجدد سرور، نیاز به راه‌اندازی دستی تامکت نباشد، فایل سرویس با محتوای زیر را ساخته و بعد از ذخیره و بستن آن، با بازخوانی[۷۲] فایل پیکربندی systemd، سرویس فعال می‌شود. این فرآیند مشابه تنظیم اجرای تامکت روی حالت Automatic در سیستم‌عامل ویندوزی است که اگر روی حالت Manual باشد، تامکت به‌طور خودکار راه‌اندازی نشده و باید به‌صورت دستی آن را استارت نمود.

```
sudo nano /etc/systemd/system/tomcat.service    # ایجاد فایل سرویس
    [Unit]                                      # محتوی فایل سرویس
    Description=Apache Tomcat 9 (GeoServer)
    After=network.target

    [Service]
    Type=forking

    User=user
    Group=user

    Environment=JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
    Environment=CATALINA_HOME=/home/user/Downloads/tomcat9

    ExecStart=/home/user/Downloads/tomcat9/bin/startup.sh
    ExecStop=/home/user/Downloads/tomcat9/bin/shutdown.sh

    [Install]
    WantedBy=multi-user.target

sudo systemctl daemon-reload                # بازخوانی فایل systemd
sudo systemctl enable tomcat                # فعال‌سازی تامکت
sudo systemctl start tomcat                 # راه‌اندازی تامکت
sudo systemctl status tomcat                # وضعیت تامکت
```

---

[۷۰] Start

[۷۱] Stop

[۷۲] Reload

۳-۱-۳- مشاهده سرویس‌های مکانی

پس از استقرار ژئوسرور روی تامکت، برای اطمینان از اجرای صحیح آن، باید با استفاده از آدرس IP سرور و پورت تامکت (مثلاً http://<IP>:port/geoserver) اتصال را بررسی کرد. اگر تامکت به‌درستی اجرا شده و ژئوسرور نیز بر روی تامکت با موفقیت مستقر شده باشد، مرورگر صفحه‌ی پیش‌فرض ژئوسرور را نمایش می‌دهند. این آزمون ساده نشان می‌دهد که هم تامکت فعال است و هم ژئوسرور روی آن در حال اجراست. نکته‌ی قابل توجه این است که بایستی دسترسی شبکه‌ای به پورت و آدرس مذکور تعریف شده باشد.

مرحله ۱۵: درخواست دسترسی به آدرس و پورت تامکت از رایانه شخصی

همانند مرحله ۱:، از واحد فناوری اطلاعات دسترسی خواندن[۷۳] به آدرس سرور و پورت تامکت (و ژئوسرور)، از مسیر رایانه شخصی درخواست می‌شود. لازم به ذکر است که پورت پیش‌فرض تامکت (و ژئوسرور) ۸۰۸۰ می‌باشد، ولی جهت اطمینان از پورت، از مسیر ذیل می‌توان پورت را مشاهده و ویرایش نمود که خروجی دستور در شکل ۳-۴ نمایش داده شده است:

```
nano ~/Downloads/tomcat9/conf/server.xml        # امکان مشاهده و ویرایش پورت
```

```
<Service name="Catalina">

  <!--The connectors can use a shared executor, you can define one or more named thread pools-->
  <!--
  <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
      maxThreads="150" minSpareThreads="4"/>
  -->

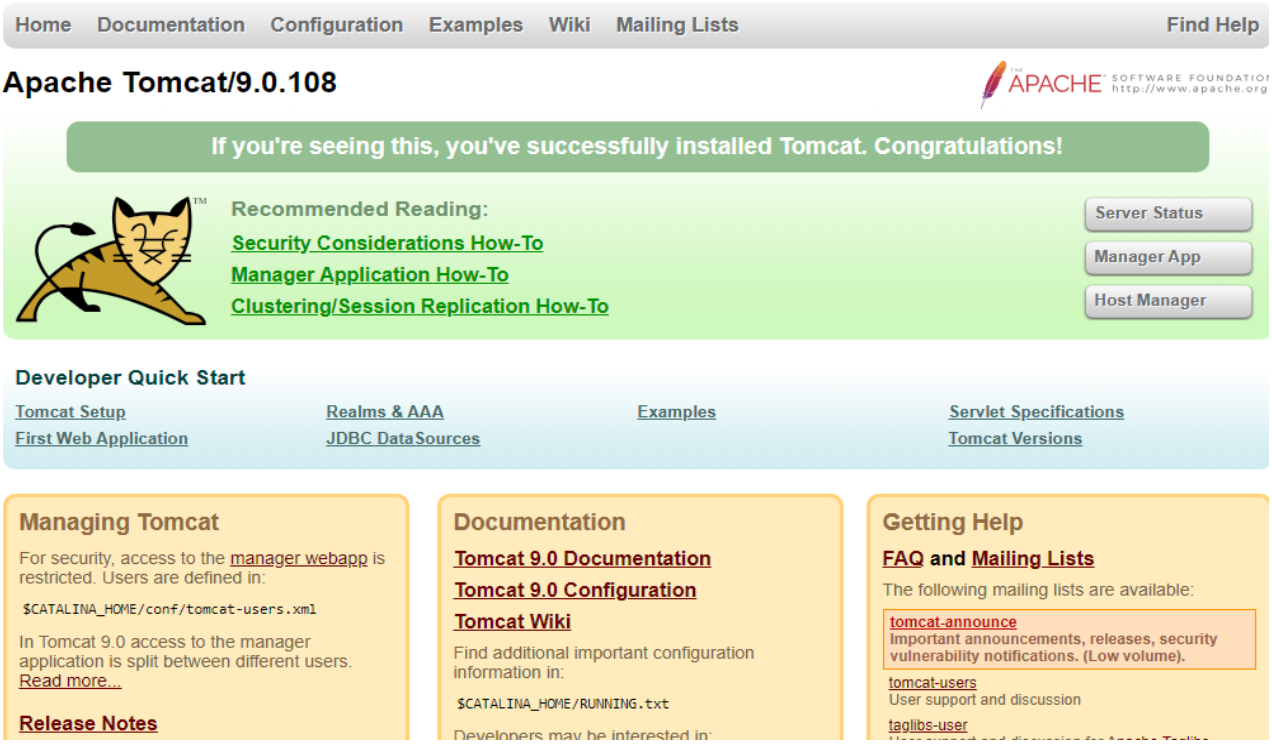

  <!-- A "Connector" represents an endpoint by which requests are received
       and responses are returned. Documentation at :
       Java HTTP Connector: /docs/config/http.html
       Java AJP  Connector: /docs/config/ajp.html
       APR (HTTP/AJP) Connector: /docs/apr.html
       Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
  -->
  <Connector port="8080" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8443"
             maxParameterCount="1000"
             />
  <!-- A "Connector" using the shared thread pool-->
  <!--
  <Connector executor="tomcatThreadPool"
             port="8080" protocol="HTTP/1.1"
             connectionTimeout="20000"
             redirectPort="8443"
             maxParameterCount="1000"
             />
  -->
  <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
       This connector uses the NIO implementation. The default
       SSLImplementation will depend on the presence of the APR/native
```

شکل ۳-٤ – مشاهده و امکان ویرایش پورت اختصاصی ژئوسرور

[۷۳] Read

مرحله ۱۶: تست نمایش تامکت

پس از اخذ دسترسی، در مرورگر رایانه شخصی، آدرس http://<IP>:Port وارد می‌شود که بایستی کاربر را به صفحه تامکت راهنمایی کند (شکل ۳-۵).



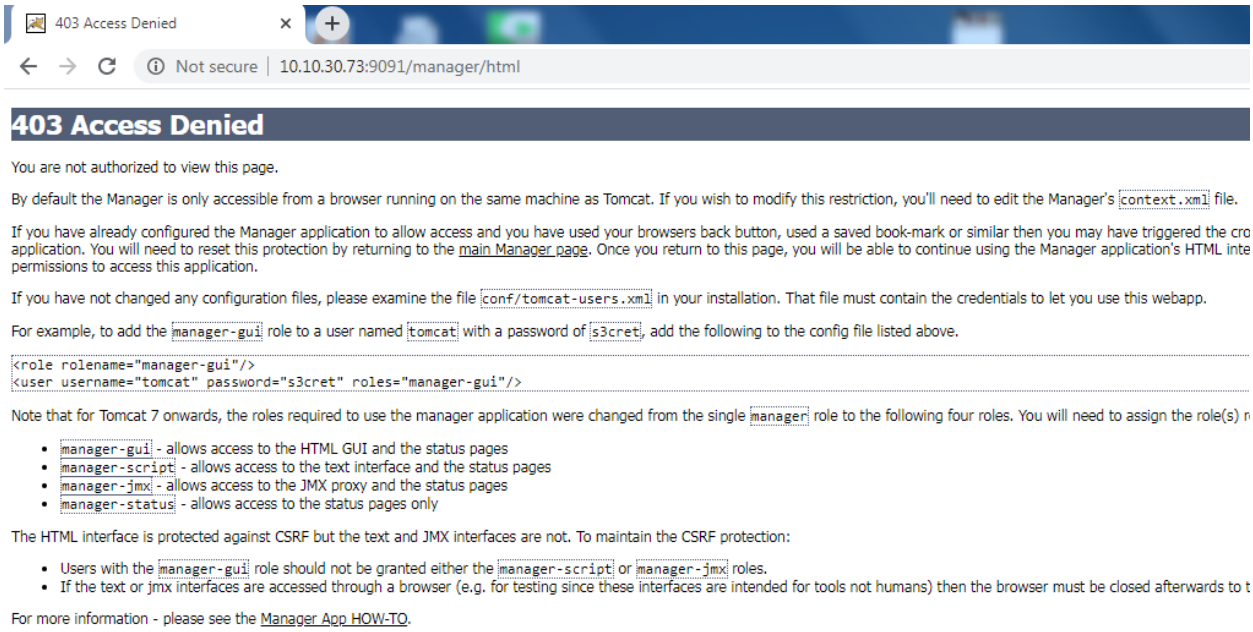
شکل ۳-۵ – نمایی از صفحه ورودی تامکت

مرحله ۱۷: دسترسی به صفحه مدیریت تامکت

برای ورود به رابط مدیریتی تامکت، از دکمه Manager App استفاده می‌شود. ممکن است در این هنگام با پیغام خطایی مشابه شکل ۳-۶ مواجه شد، شایان ذکر است هنگام دسترسی به تامکت از داخل سرور، این خطا مشاهده نمی‌شود، اما در صورتی که هدف، اتصال از یک رایانه دیگر به تامکت باشد، پیغام مذکور ظاهر خواهد شد. علت این موضوع آن است که به‌منظور افزایش امنیت، دسترسی به رابط مدیریتی تامکت به‌صورت پیش‌فرض فقط از داخل خود سرور مجاز است. برای رفع این محدودیت نیاز است تا در فایل context.xml مجموعه عبارات valve calssName به‌صورت کامنت درآمده و غیر قابل اجرا شوند (شکل ۳-۷). بدین‌منظور بایستی دستور زیر وارد شده و خطوط مذکور کامنت شوند:

```
nano ~/Downloads/tomcat9/webapps/manager/META-INF/context.xml
```

پس از اعمال تغییرات، با فشردن کلیدهای Ctrl+X از ویرایشگر خارج شده و با انتخاب y (yes) تغییرات ذخیره می‌شوند. پس از این ویرایش، برنامه مدیریت تامکت از سایر رایانه‌های مجاز شبکه نیز قابل دسترسی خواهد بود.



شکل ۳-۶ – خطای نمایش صفحه تامکت پیش از ویرایش فایل context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<Context antiResourceLocking="false" privileged="true" >
  <!--  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
    <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.catalina
    -->
</Context>
```

شکل ۳-۷ – ویرایش فایل context.xml

مرحله ۱۸: ورود به صفحه مدیریت تامکت

به‌هنگام فراخوانی صفحه مدیریت تامکت، نام کاربری و رمز عبور درخواست می‌شود. این اطلاعات در فایل tomcat-users.xml تعریف می‌شوند و در نصب پیش‌فرض تامکت معمولاً کاربر مدیریتی فعال وجود ندارد. در صورتی که نام کاربری و رمز عبور از قبل تعریف نشده باشد، یا نیاز به تغییر آن‌ها وجود داشته باشد، می‌توان فایل زیر را ویرایش کرد:

```
nano ~/Downloads/tomcat9/conf/tomcat-users.xml        # امکان مشاهده و ویرایش نام کاربری و رمز عبور
```

برای ایجاد کاربر با دسترسی به رابط مدیریتی تامکت، می‌توان ورودی زیر را در بخش مربوط به کاربران اضافه یا ویرایش کرد:

تعریف/ویرایش نام کاربری و رمز عبور #

```
<user username="*" password="*" roles="manager-gui"/>
```

```
  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
<user username="admin" password="admin" roles="manager-gui"/>
<!--
  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary.

  Built-in Tomcat manager roles:
    - manager-gui    - allows access to the HTML GUI and the status pages
    - manager-script - allows access to the HTTP API and the status pages
    - manager-jmx    - allows access to the JMX proxy and the status pages
```

شکل ۳-۸ – تعریف یا ویرایش نام کاربری و رمز عبور برای مدیریت تامکت

پس از ویرایش، با فشردن دکمه ctrl+x خارج و با y(yes) تغییرات ذخیره می‌شوند. در صورت نیاز، تامکت با استفاده از فایل‌های راه‌اندازی و توقف موجود در پوشه bin، مطابق مرحله ۱۳، مجدداً اجرا شده و با نام کاربری و رمز عبور تعریف شده، وارد رابط مدیریتی تامکت خواهد شد (شکل ۳-۹):

Tomcat Web Application Manager

Message:	OK

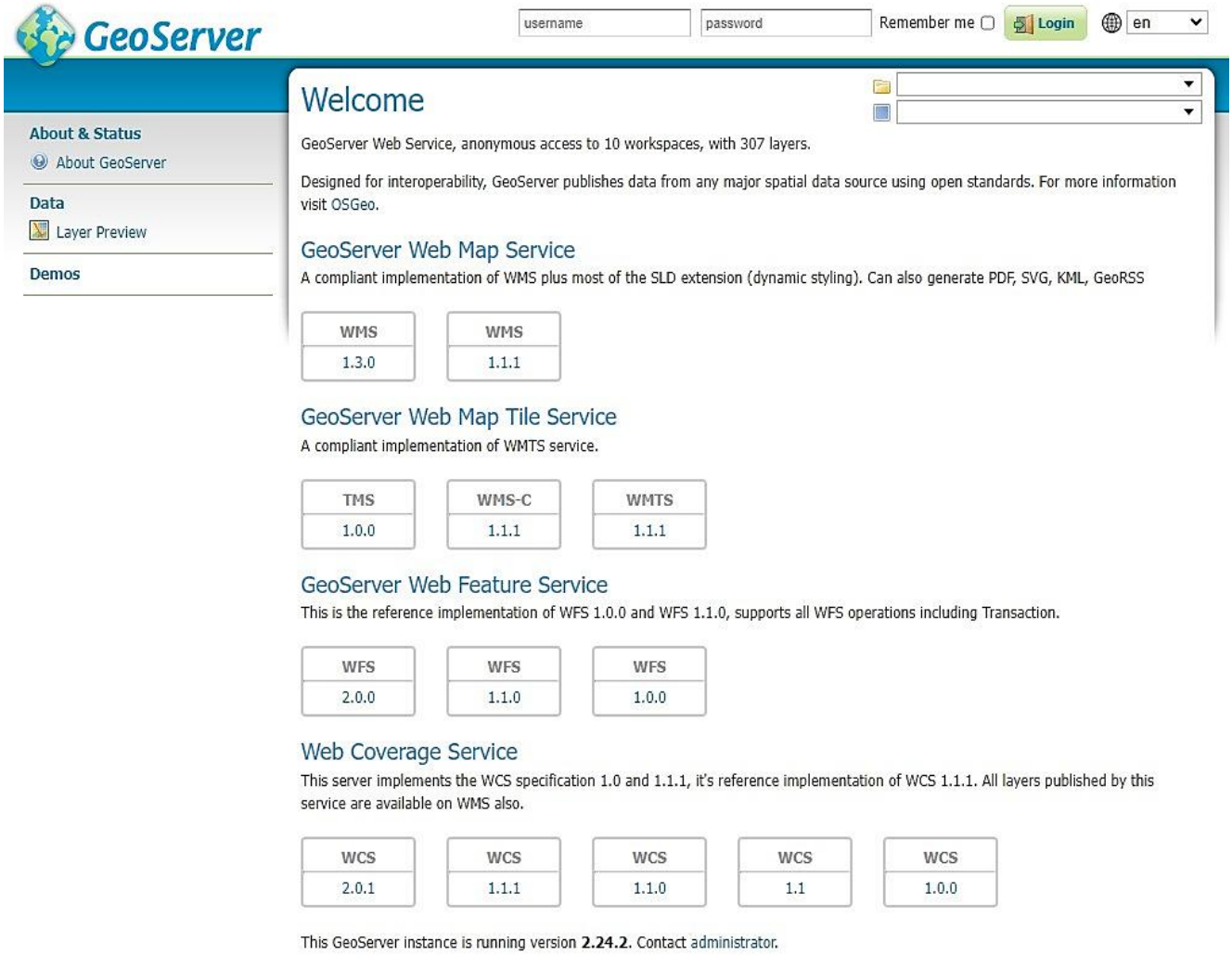
Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
					Start Stop Reload Undeploy

شکل ۳-۹ – نمایی از صفحه مدیریت تامکت

مرحله ۱۹: تست نمایش ژئوسرور

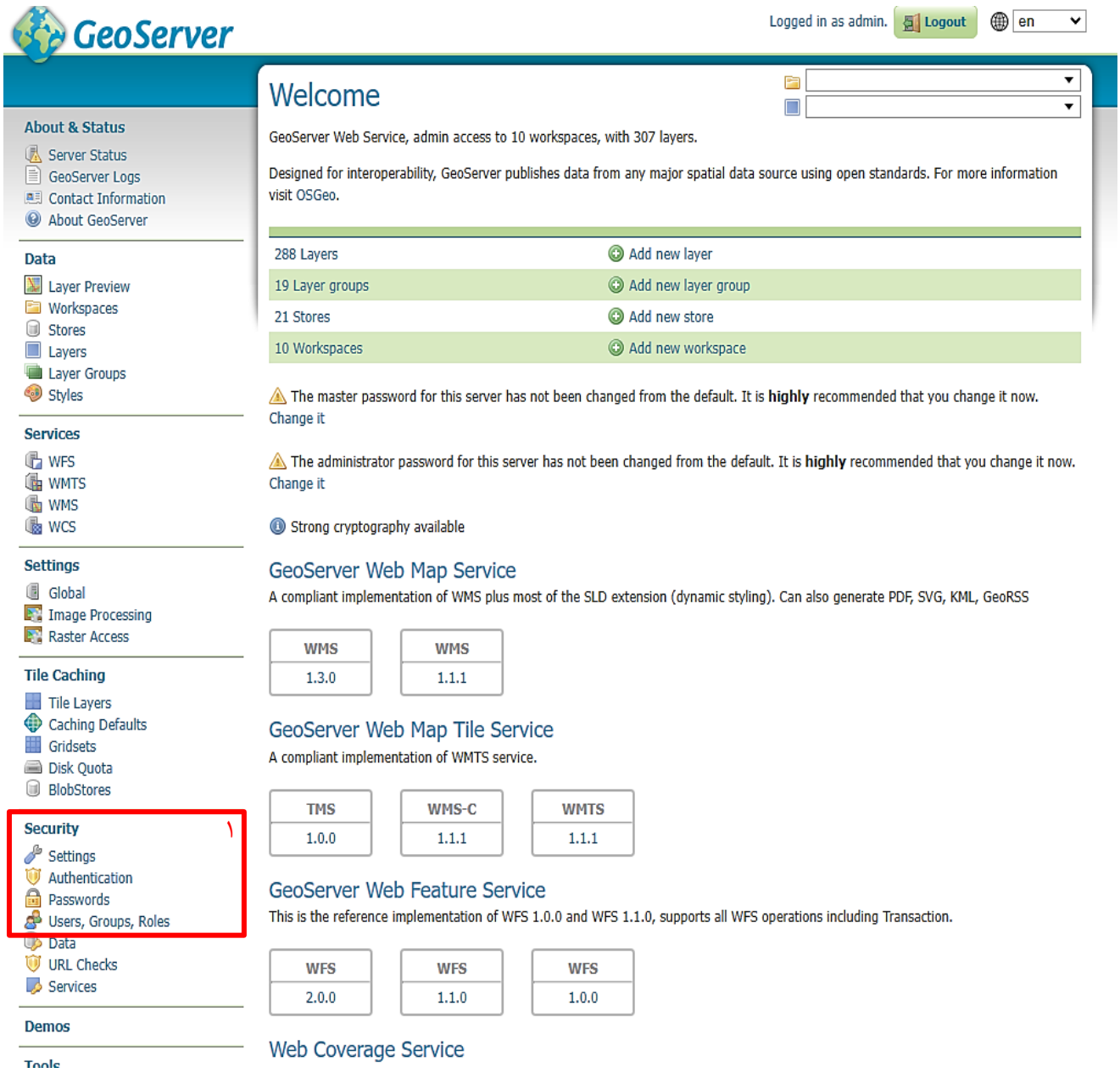
برای دسترسی به ژئوسرور، در مرورگر رایانه شخصی با آدرس http://<IP>:Port/geoserver وارد شده و یا از پنجره رابط مدیریتی تامکت با کلیک روی geoserver/ ، صفحه ژئوسرور نمایان خواهد شد (شکل ۳-۱۰):



شکل ۳-۱۰ – نمایی از صفحه اولیه ژئوسرور

مرحله ۲۰: ورود به پنجره مدیریت ژئوسرور

پس از ورود به رابط کاربری ژئوسرور با نام کاربری و رمز عبور، منوهای مربوط به تنظیمات، مدیریت و نمایش سرویس‌ها و سایر ابزارهای ژئوسرور در اختیار کاربر قرار می‌گیرند (شکل ۳-۱۱). نام کاربری پیش‌فرض admin و رمز عبور پیش‌فرض geoserver می‌باشند. البته این مقادیر در صورت نیاز، از طریق بخش تنظیمات در صفحه وب ژئوسرور (کادر شماره ۱ شکل ۳-۱۱) به‌راحتی قابل تغییر هستند.



شکل ۳-۱۱ – نمایی از ابزارهای ژئوسرور پس از ورود کاربر به آن

مرحله ۲۱: پیش‌نمایش سرویس‌های مکانی در ژئوسرور

اگر در رابط کاربری ژئوسرور، از قسمت ابزارهای سمت چپ پنجره، گزینه Layer Preview انتخاب شوند، فهرست لایه‌ها و سرویس‌های مکانی موجود نمایش داده می‌شود. این بخش امکان مشاهده یا دریافت خروجی لایه‌ها در قالب‌های مختلفی مانند OpenLayers، GML و KML را فراهم می‌کند. از میان این گزینه‌ها، OpenLayers امکان نمایش مستقیم لایه در مرورگر را فراهم می‌سازد.

سایر بخش‌های مدیریتی ژئوسرور، از جمله Workspaces، Stores و Layers، برای ایجاد و مدیریت فضاهای کاری، منابع داده، لایه‌ها و سرویس‌های مکانی مورد استفاده قرار می‌گیرند.

۳-۱-۴- راه‌اندازی سرویس‌های مکانی از پیش موجود

از آنجا که تعداد سرویس‌های مکانی سازمان‌ها از جمله سازمان نقشه‌برداری کشور بسیار زیاد بوده و این سرویس‌ها پیش‌تر مطابق استانداردها و ضوابط مصوب طراحی و پیکربندی شده‌اند، نیازی به ایجاد مجدد آن‌ها وجود ندارد. با استفاده از نسخه پشتیبان پیکربندی‌ها، داده‌ها و تنظیمات موجود و بازیابی آن‌ها در ژئوسرور مستقر بر روی سرور هدف، می‌توان از تکرار فرایند ایجاد و پیکربندی سرویس‌ها جلوگیری کرد.

برای نمونه، فایل‌های پشتیبان موجود در اختیار اداره سامانه‌های مکانی سازمان نقشه‌برداری کشور برای لایه‌های اطلاعاتی ۱:۲۵۰۰۰ عبارتند از:

```
Geo_Base_Final          # دایرکتوری حاوی داده‌ها و تنظیمات با حجم بالا
Geodatabase.ttf         # فایل استایل و فونت
LargeScale.ttf          # فایل استایل و فونت
LargeScale-t.ttf        # فایل استایل و فونت
```

همچنین یک گروه‌لایه کلی نیز وجود دارد که تمامی لایه‌های مقیاس ۱:۲۰۰۰ در آن تعریف شده‌اند.

مرحله ۲۲: آماده‌سازی ژئوسرور برای بارگذاری سرویس‌های موجود

پیش از شروع بارگذاری سرویس‌های مکانی سازمان (بازیابی از نسخه پشتیبان) بهتر است تامکت را از پوشه bin متوقف کرد تا ژئسرور در زمان جایگزینی فایل‌ها در حال اجرا نباشد. سپس برای اطمینان از جایگزینی کامل فایل‌های پشتیبان با فایل‌های پیش‌فرض ژئوسرور، پوشه‌ی زیر به‌همراه فایل‌های داخل آن حذف می‌شوند:

```
cd ~/Downloads/tomcat9/bin                                  # قرار گرفتن در مسیر هدف
sudo ./shutdown.sh                                          # توقف تامکت
sudo rm -r ~/Downloads/tomcat9/webapps/geoserver/data/styles # حذف پوشه استایل نصب‌شده بهمراه فایل‌ها
```

مرحله ۲۳: بارگذاری سرویس‌ها و استایل‌های موجود روی ژئوسرور

با توجه به حجم بالای داده‌ها، ابتدا فایل‌ها و پوشه‌های موردنیاز در یک مسیر موقت بر روی سرور هدف قرار می‌گیرند. سپس، مطابق با ساختار موردنیاز برای راه‌اندازی سرویس‌های مکانی، به مسیرهای مشخص‌شده در ادامه منتقل می‌شوند:

```
sudo cp -r /home/Geo_Base_Final/data/data/ShpLevelFinal ~/Downloads/tomcat9/webapps/geoserver/data/data
                                                            # کپی پوشه شیپ‌فایل‌ها در مسیر هدف

sudo cp -r /home/Geo_Base_Final/data/styles ~/Downloads/tomcat9/webapps/geoserver/data
                                                            # کپی پوشه styles در مسیر هدف

sudo cp -r /home/Geo_Base_Final/data/workspaces/ShpLevelFinal_WS ~/Downloads/tomcat9/webapps/geoserver/data/workspaces
                                                            # کپی پوشه workspace مربوطه در مسیر هدف

sudo cp -r /home/Geo_Base_Final/*.ttf ~/Downloads/tomcat9   # کپی یکجای هر سه فایل استایل و فونت
```

مرحله ۲۴: مجوز دسترسی به فایل‌های بارگذاری شده

پس از قرار دادن فایل‌ها در مسیرهای مشخص شده، لازم است مالکیت و مجوزهای دسترسی مسیر تامکت به‌گونه‌ای تنظیم شود که کاربر اجراکننده‌ی تامکت، بتواند فایل‌ها و پوشه‌های مورد نیاز را بخواند و در صورت نیاز در آن‌ها تغییر ایجاد کند:

```
sudo chmod -R 755 ~/Downloads/tomcat9
sudo chown -R user:user ~/Downloads/tomcat9
```

 # کلمات user با نام کاربری شخصی جایگزین شود

مرحله ۲۵: تست پیش‌نمایش سرویس‌های مکانی

پس از انجام مراحل قبل و راه‌اندازی مجدد تامکت از پوشه bin، با ورود به ژئوسرور بایستی لایه‌ها و سرویس‌های مکانی مربوط به شیپ‌فایل‌های بارگذاری شده، در فهرست لایه‌ها مشاهده شود.

در صورت پیش‌نمایش این لایه‌ها از طریق OpenLayers، ممکن است صفحه‌ای سفید و بدون نمایش عوارض ظاهر شود. در این حالت، وجود لایه‌ها در فهرست ژئوسرور نشان می‌دهد که داده‌ها و پیکربندی کلی آن‌ها بارگذاری شده‌اند، اما نمایش عوارض با مشکل مواجه است. در این مستند، این مشکل ناشی از تنظیمات استایل و فایل‌های فونت مرتبط با آن‌هاست که در مرحله بعد اصلاح می‌شود.

مرحله ۲۶: رفع ایراد فراخوانی استایل‌ها از مسیرهای نسبی تعریف شده

فایل‌های استایل در مسیر tomcat9/webapps/geoserver/data/styles و در قالب فایل‌های SLD تعریف شده‌اند که در برخی از این فایل‌ها، برای نمایش نمادها از فونت‌های اختصاصی استفاده شده و مسیر فایل فونت به‌صورت نسبی تعریف شده است؛ برای مثال، تعدادی از سبک‌های مربوط به لایه‌های نقطه‌ای به فایل Geodatabase.ttf ارجاع می‌دهند. بنابراین لازم است این مسیر نسبی به مسیر مطلق فایل فونت روی سرور تغییر یابد. به این منظور، دستور زیر تمام ارجاعات مربوط به Geodatabase.ttf را در فایل‌های SLD به مسیر صحیح و مطلق آن تغییر می‌دهد::

```
sudo find ~/Downloads/tomcat9/webapps/geoserver/data/styles -name "*.sld" -exec sed -i
's|ttf:///Geodatabase.ttf|ttf:///home/user/Downloads/tomcat9/Geodatabase.ttf|g' {} \;
```

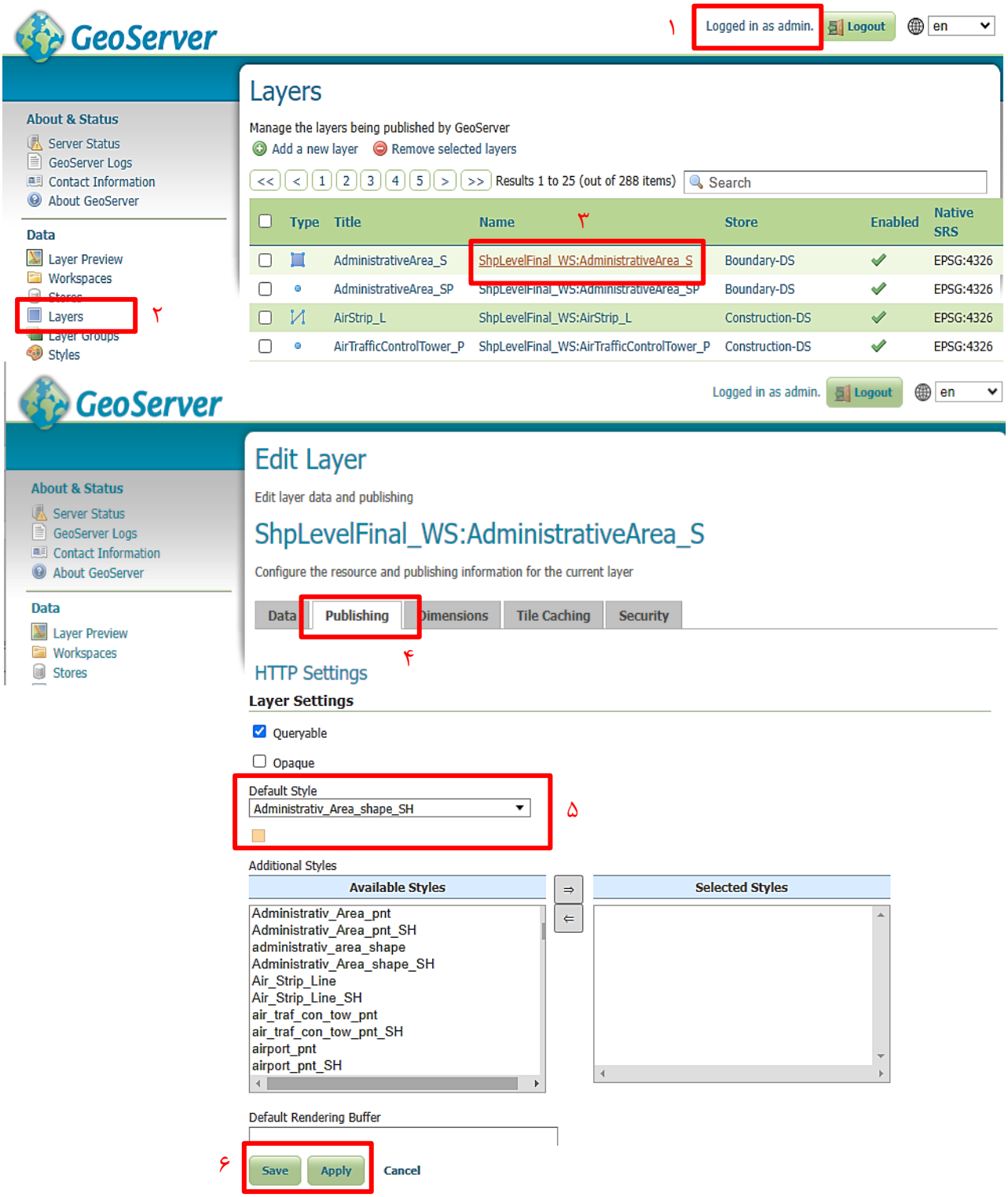
 # کلمه user با نام کاربری شخصی جایگزین شود

پس از اعمال این تغییر، لایه‌ها در محدوده مقیاس‌های تعریف‌شده در فایل‌های SLD به‌درستی نمایش داده می‌شوند. با این حال، ممکن است در برخی سطوح بزرگ‌نمایی یا کوچک‌نمایی، عوارض همچنان نمایش داده نشوند که علت آن به تنظیمات مقیاس در استایل‌های تعریف‌شده مربوط است و در مرحله بعد بررسی و اصلاح می‌شود.

مرحله ۲۷: رفع محدودیت مقیاس نمایشی تعریف شده

با بررسی فایل‌های SLD در پوشه استایل‌ها مشخص می‌شود که برای اغلب لایه‌ها دو استایل با نام‌های *.sld و *_SH.sld تعریف شده است. استایل‌های *.sld برای لایه‌هایی تهیه شده‌اند که درون گروه‌لایه نمایش داده می‌شوند و دارای محدودیت مقیاس نمایش هستند؛ این محدودیت باعث می‌شود نمایش گروه‌لایه بهینه‌تر، سریع‌تر و از نظر بصری مناسب‌تر باشد. در مقابل، استایل‌های *_SH.sld برای نمایش مستقل همان لایه‌ها خارج از گروه‌لایه استفاده می‌شوند و محدودیت مقیاس نمایشی ندارند.

برای نمایش پیش‌فرض لایه‌ها بدون محدودیت مقیاس نمایش، می‌توان با استفاده از ابزار خود ژئوسرور و همانند شکل ۳-۱۲ از بخش Publishing، استایل پیش‌فرض هر لایه را بر روی *_SH.sld گذاشت تا پیش‌نمایش لایه در OpenLayers در همه سطوح بزرگ‌نمایی، عوارض را نمایش دهد.



شکل ۳-۱۲ – تنظیمات استایل پیش‌فرض لایه‌ها

با انجام مراحل فوق، سرویس‌های مکانی از پیش‌موجود سازمان بر روی سرور هدف راه‌اندازی شده و از طریق ژئوسرور قابل مشاهده و بهره‌برداری خواهند بود. برای نگهداری و بازیابی این تنظیمات در آینده، لازم است از تمامی لایه‌ها، سبک‌ها، داده‌ها و تنظیمات انجام‌شده، نسخه پشتیبان استاندارد تهیه گردد.

مرحله ۲۸: تهیه نسخه پشتیبان

در بروز خطا در سامانه، خرابی سخت‌افزار یا نیاز به انتقال ژئوسرور به سرور جدید، مهم‌ترین راه برای بازگردانی سریع تنظیمات و سرویس‌ها استفاده از نسخه پشتیبان[۷۴] است. به این منظور، ابتدا باید مسیر اصلی داده‌ها و تنظیمات ژئوسرور مشخص شود. این مسیر معمولاً در محل استقرار ژئوسرور قرار دارد؛ در این مستند:

home/*user*/Downloads/tomcat9/webapps/geoserver/data

با دستور زیر، از این پوشه نسخه پشتیبان فشرده تهیه شده و با نامی شامل تاریخ روز در پوشه backup ذخیره می‌شود:

```
sudo tar -czvf /home/user/backup/geoserver_backup_$(date +%Y-%m-%d).tar.gz
/home/user/Downloads/tomcat9/webapps/geoserver/data
```

 # کلمات user با کاربری شخصی جایگزین شود

توصیه می‌شود فایل نسخه پشتیبان پس از ایجاد، به یک سیستم یا فضای ذخیره‌سازی دیگر منتقل شود تا در صورت بروز مشکل در سرور پایش‌شونده، نسخه پشتیبان نیز از بین نرود.

مرحله ۲۹: اتوماتیک‌سازی بکاپ‌گیری

برای اتوماتیک‌سازی فرآیند تهیه نسخه پشتیبان و حذف نیاز به اجرای دستی مرحله قبل، می‌توان در بازه‌های زمانی مشخص، مانند هفتگی یا ماهانه، از تنظیمات، داده‌ها و لایه‌های مکانی نسخه پشتیبان تهیه کرد. به این ترتیب، در صورت بروز خطا یا خرابی، آخرین نسخه سالم در دسترس خواهد بود. برای این منظور، ابتدا اسکریپتی شامل دستورات لازم برای فشرده‌سازی و ذخیره مسیرهای ضروری تهیه می‌شود. همچنین برای مدیریت فضای ذخیره‌سازی سرور، در هر بار اجرای اسکریپت، نسخه‌های پشتیبان قدیمی‌تر از ۱۲۰ روز حذف می‌گردند.

```
nano backup_geoserver.sh                                  # ساخت فایل اسکریپ با محتوای زیر

#!/bin/bash
BACKUP_DIR="/home/user/backup"                            # کلمه user با نام کاربری جایگزین شود
SRC_DIR="/home/user/Downloads/tomcat9/webapps/geoserver/data"
DATE=$(date +%Y-%m-%d)
FILENAME="geoserver_backup_$DATE.tar.gz"

mkdir -p "$BACKUP_DIR"                                    # تعیین نام فایل بر اساس تاریخ
tar -czf "$BACKUP_DIR/$FILENAME" --exclude='gwc' -C       # ایجاد پوشه ذخیره‌سازی فایل بکاپ
"$SRC_DIR" .                                              # فشرده‌سازی و ذخیره فایل‌های بکاپ
find "$BACKUP_DIR" -type f -name "geoserver_backup_*.tar.gz" -
mtime +120 -exec rm {} \;                                 # حذف بکاپ‌های قدیمی‌تر از ۱۲۰ روز
```

[۷۴] Backup

پس از ذخیره و بستن فایل اسکریپ، مجوز اجرای فایل تنظیم می‌گردد:

```
chmod +x backup_geoserver.sh          # اعطای مجوز اجرا
```

برای زمان‌بندی خودکار فرایند تهیه نسخه پشتیبان، فایل کران‌تب[۷۵] ویرایش می‌شود. به این ترتیب می‌توان اجرای اسکریپت را در زمان‌های مشخص برنامه‌ریزی کرد. برای مثال، دستور زیر طوری تنظیم می‌شود که اسکریپت در روز اول هر ماه، ساعت ۰۸:۰۰ صبح، اجرا شده و نسخه پشتیبان تهیه گردد:

```
sudo crontab -e                       # باز کردن فایل تنظیمات
```

پس از اجرای دستور، با انتخاب گزینه‌ی ۱ فایل زمان‌بندی در ویرایشگر nano باز شده و سطر زیر به آن اضافه می‌شود.

```
0 8 1 * * /home/user/backup_geoserver.sh   # روز ۱ هرماه روی ساعت ۸ و دقیقه‌ی ۰ اجرا می‌شود
```

کلمه user با نام کاربری جایگزین شود

۳-۲- آماده‌سازی سرور پایش‌شونده جهت پایش

وقتی یک سامانه‌ی نرم‌افزاری به مرحله‌ی بهره‌برداری می‌رسد، کار توسعه و راه‌اندازی به پایان نمی‌رسد؛ بلکه تازه چالش اصلی یعنی نگهداری و پایش مداوم آغاز می‌شود. در سرورهایی که سرویس‌های مکانی (مانند ژئوسرور) بر بستر تامکت و جاوا اجرا می‌شوند، تضمین پایداری و کارایی بدون ابزارهای پایش تقریباً غیرممکن است. پایش به معنای گردآوری، ذخیره‌سازی و تحلیل داده‌های مربوط به منابع سیستمی و رفتار سرویس‌هاست؛ داده‌هایی که با آن‌ها می‌توان سلامت، عملکرد و روند تغییرات یک سامانه را در طول زمان دنبال کرد.

هدف از پایش صرفاً جمع‌آوری داده نیست، بلکه ایجاد دیدی جامع و بلادرنگ نسبت به وضعیت سیستم است. مدیر سیستم باید بتواند مصرف پردازنده، حافظه، دیسک، شبکه و وضعیت سرویس‌های حیاتی را به‌صورت مستمر بررسی کند. اهمیت پایش در سرویس‌های مکانی دوچندان است، زیرا این سرویس‌ها معمولاً حجم زیادی از داده‌های گرافیکی و مکانی را پردازش می‌کنند و بار پردازشی و مصرف منابع آن‌ها ممکن است به‌صورت ناگهانی افزایش یابد.

به همین دلیل در اکوسیستم مانیتورینگ مدرن، ابزارهای پایش با لایه‌بندی مشخص طراحی شده‌اند. در این معماری‌ها تقسیم وظایف بین سرور پایش‌کننده و سرورهای پایش‌شونده یک اصل کلیدی است. سرور پایش‌کننده محیطی متمرکز برای اجرای ابزارهای جمع‌آوری، ذخیره و نمایش داده‌ها فراهم می‌کند؛ و ترجیحا ابزارهای GeoHealthCheck و Prometheus و Grafana بر روی این سرور پایش‌کننده نصب و اجرا می‌شوند.

[۷۵] Crontab: فایل زمان‌بندی وظایف در سیستم‌های شبه‌یونیکس است که اجرای خودکار دستورها یا اسکریپت‌ها را در زمان‌های مشخص کنترل می‌کند.

در سمت سرورهای پایش‌شونده، Exporterها نصب می‌شوند تا داده‌ها را مستقیماً از منابع جمع‌آوری کنند. برای مثال، Node Exporter معیارهای سخت‌افزاری و سیستم‌عامل را استخراج می‌کند، و Tomcat Exporter معیارهای عملکردی تامکت و ماشین مجازی جاوا (ژئوسرور بر روی آن اجرا می‌شود) را جمع‌آوری می‌کند؛ از جمله مصرف حافظه[۷۶] و نشست‌های فعال[۷۷] کاربران. این کار باعث می‌شود Prometheus بتواند داده‌های دقیق و به‌روز را از Exporterها جمع‌آوری کند، بدون اینکه فشار زیادی بر سرور اصلی وارد شود. این تقسیم وظایف دو مزیت اصلی دارد:

- تمرکز ابزارهای تحلیل و نمایش روی یک سرور پایش‌کننده، که مدیریت و نگهداری آن ساده‌تر است.
- جمع‌آوری نزدیک به منبع داده‌ها با نصب Exporterها روی سرورهای پایش‌شونده، که باعث گردآوری مستقیم و قابل اعتماد داده‌ها از منبع می‌شود.

۳-۲-۱- نصب Node-Exporter

Node Exporter یکی از Exporterهای کلیدی در اکوسیستم Prometheus است که وظیفه گردآوری معیارهای سیستم‌عامل لینوکس و سخت‌افزار را بر عهده داشته و آن‌ها را برای جمع‌آوری توسط Prometheus در دسترس قرار می‌دهد. این ابزار به‌صورت سبک و کارآمد طراحی شده و اطلاعات دقیقی از وضعیت منابع سرور در اختیار مدیر سیستم قرار می‌دهد. معیارهایی که Node Exporter ارائه می‌دهد شامل مصرف پردازنده، حافظه، فضای دیسک، وضعیت شبکه، میانگین بار سیستم و بسیاری از شاخص‌های حیاتی دیگر است. این معیارها به مدیر سیستم کمک می‌کنند تا سلامت سرورهای پایش‌شونده را بدون وارد کردن بار قابل توجه به سیستم بررسی کند.

Node Exporter به‌ویژه در محیط‌های مکانی اهمیت دارد، زیرا ابزارهایی مانند GeoServer ممکن است هنگام پردازش درخواست‌های نقشه، مصرف پردازنده، حافظه، دیسک یا شبکه را به‌صورت ناگهانی افزایش دهند. با جمع‌آوری معیارها در سطح سیستم، مدیر سیستم می‌تواند الگوهای مصرف منابع را شناسایی کند، هشدارهای پیشگیرانه تعریف نماید و مشکلاتی مانند پر شدن حافظه، افزایش بار سیستم، کمبود فضای دیسک یا اختلالات شبکه را پیش از اثرگذاری جدی بر سرویس‌های مکانی تشخیص دهد.

از مزایای این Exporter می‌توان به سبک بودن، کارایی مناسب، ارائه معیارهای دقیق و قابل اعتماد، امکان استفاده در داشبوردهایی مانند Grafana و پشتیبانی از تعریف هشدار در Prometheus اشاره کرد. در کنار این مزایا، محدودیت‌هایی نیز وجود دارد، از جمله:

- محدود به سطح سیستم: Node Exporter فقط وضعیت منابع سخت‌افزاری و سیستم‌عامل را پایش می‌کند و اطلاعات دقیق مربوط به خود برنامه‌های کاربردی (مانند ژئوسرور) را ارائه نمی‌دهد.

---

[۷۶] Heap Memory

[۷۷] Session

- نیاز به تنظیمات امنیتی تکمیلی: معیارهای جمع‌آوری‌شده به‌صورت پیش‌فرض از طریق یک آدرس HTTP در دسترس قرار می‌گیرند و معمولاً سازوکار داخلی احراز هویت یا رمزنگاری ندارند. بنابراین لازم است دسترسی به آن‌ها از طریق تنظیمات شبکه، فایروال یا سایر لایه‌های امنیتی محافظت شود.

در ادامه مراحل نصب و راه‌اندازی Node Exporter با جزئیات تشریح می‌گردد.

مرحله ۱: دانلود بسته نرم‌افزاری

با توجه به اینکه دسترسی اغلب سرورها به اینترنت آزاد بنا به دلایل امنیتی محدود است، بایستی فایل نصبی ابزار را در سیستم دارای اینترنت دانلود نموده و سپس به سرور منتقل کرد. برای دانلود می‌توان به لینک زیر مراجعه نمود:

https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz

مرحله ۲: انتقال فایل دانلود شده به سرور

فایل دانلود شده از مرحله قبل بر روی سرور پایش‌شونده و داخل پوشه‌ای مشخص (همانند Downloads) همانند مرحله ۶ بخش پیشین، بارگذاری می‌گردد.

مرحله ۳: نصب Node Exporter

برای نصب فایل دانلود شده، وارد پوشه حاوی آن شده و با استفاده از دستورات زیر از حالت فشرده خارج نموده و فایل اجرایی به پوشه bin کاربر منتقل می‌شود:

```
cd Downloads                                          # ورود به پوشه حاوی فایل نصب
tar -xvf node_exporter-1.8.2.linux-amd64.tar.gz       # خارج کردن از حالت فشرده
sudo cp ~/Downloads/node_exporter/node_exporter       # کپی فایل اجرایی در یکی از مسیرهای
/usr/local/bin/node_exporter                          استاندارد فایل‌های اجرایی
```

مرحله ۴: مدیریت دسترسی به فایل اجرایی Node Exporter

برای اجرای بدون نقص فایل مذکور، با دستور زیر، مجوز اجرا اعطا می‌شود:

```
sudo chmod +x /usr/local/bin/node_exporter
```

مرحله ۵: ایجاد فایل systemd جهت اجرای اتوماتیک

به منظور اینکه Node Exporter در هر لحظه و پس از هر بار راه‌اندازی مجدد سیستم، بدون نیاز به تنظیم و اجرای دستی، بصورت اتوماتیک در کاربر مشخص‌شده اجرا شود، ابتدا فایل node_exporter.service در مسیر systemd با محتویات زیر ساخته می‌شود و در پایان بایستی با فشردن دکمه ctrl+x و y محتویات را ذخیره و خارج شد:

```
sudo nano /etc/systemd/system/node_exporter.service
[Unit]
Description=Prometheus Node Exporter
Wants=network-online.target
```

```
After=network-online.target
[Service]
User=user
Group=user
Type=simple
ExecStart=/usr/local/bin/node_exporter
[Install]
WantedBy=multi-user.target
```

مرحله ۶: بازخوانی systemd و فعال‌سازی سرویس

برای اینکه systemd فایل سرویس ایجادشده در مرحله قبل را شناسایی کند، ابتدا باید پیکربندی آن بازخوانی شود. سپس سرویس Node Exporter فعال و اجرا می‌گردد. فعال‌سازی سرویس باعث می‌شود Node Exporter پس از هر بار راه‌اندازی مجدد سرور نیز به‌صورت خودکار اجرا شود. به این منظور، دستورات زیر به‌ترتیب در محیط لینوکس اجرا می‌شوند:

```
sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
```

مرحله ۷: بررسی وضعیت و تست راه‌اندازی سرویس

پس از اجرای مرحله قبل، Node Exporter آماده است تا متریک‌های سرور را از طریق پورت پیش‌فرض ۹۱۰۰ در اختیار Prometheus قرار دهد. برای اطمینان از صحت عملکرد سرویس، خروجی دستورات زیر بررسی می‌شود. در صورت اجرای صحیح، وضعیت سرویس باید در حالت فعال نمایش داده شود و درخواست متریک‌ها نیز خروجی متنی شامل خطوط متریک، معمولاً با پیشوند node_ و توضیحات # HELP و # TYPE ، بازگرداند:

```
systemctl status node_exporter -l
curl http://localhost:9100/metrics | head
```

لازم به ذکر است که پورت پیش‌فرض Node Exporter برابر با ۹۱۰۰ است.

تذکر ۱:

اگر هدف، اجرای دستی Node Exporter باشد و نیازی به اجرای خودکار آن پس از راه‌اندازی مجدد سرور وجود نداشته باشد، پس از خارج کردن فایل دانلودشده از حالت فشرده، می‌توان مراحل مربوط به ایجاد فایل systemd و فعال‌سازی سرویس را انجام نداد. در این حالت، وارد پوشه استخراج‌شده شده و Node Exporter را با دستور زیر اجرا نمود:

```
cd ~/Downloads/node_exporter-1.8.2.linux-amd64
./node_exporter
```

۳-۲-۲- نصب Tomcat-Exporter

این Exporter یکی از Exporterهای تخصصی برای جمع‌آوری معیارهای عملکردی تامکت و ماشین مجازی جاوا است که معیارهای مربوط به عملکرد و سلامت آنها را جمع‌آوری کرده و آن‌ها را برای دریافت توسط Prometheus در دسترس قرار می‌دهد. به کمک این داده‌ها می‌توان وضعیت محیطی را که برنامه‌های جاوایی، از جمله ژئوسرور، روی آن اجرا می‌شوند، به‌صورت دقیق پایش کرد.

Tomcat Exporter اطلاعات مهمی از وضعیت محیط اجرای ژئوسرور ارائه می‌دهد؛ از جمله تعداد رشته‌های پردازشی، میزان مصرف حافظه‌ی ماشین مجازی جاوا و تعداد نشست‌های فعال. این اطلاعات به مدیر سیستم کمک می‌کند مشکلاتی را که ممکن است بر عملکرد سرویس‌های مکانی اثرگذار باشند، پیش از تبدیل شدن به اختلال جدی شناسایی و مدیریت کند.

در کنار Node Exporter و GeoHealthCheck، Tomcat Exporter لایه‌ی میانی پایش را تشکیل می‌دهد: Node Exporter وضعیت منابع سخت‌افزاری و سیستم‌عامل را گزارش می‌کند، GeoHealthCheck در دسترس بودن و پاسخ‌دهی سرویس‌های مکانی را بررسی می‌کند، Tomcat Exporter معیارهای عملکردی تامکت و ماشین مجازی جاوا را ارائه می‌دهد. ترکیب این سه ابزار، دیدی جامع و بلادرنگ از وضعیت سرور و سرویس‌های مکانی فراهم می‌کند.

از مزایای Tomcat Exporter می‌توان به ارائه معیارهای دقیق از عملکرد تامکت، یکپارچگی با Prometheus و Grafana و کمک به شناسایی زودهنگام مشکلات عملکردی اشاره کرد. در کنار این مزایا، محدودیت‌هایی نیز وجود دارد، از جمله:

- محدود به تامکت و برنامه‌های مبتنی بر JVM: بنابراین برای پایش سرویس‌های غیرجاوایی مناسب نیست.
- نیاز به تنظیمات امنیتی: معیارهای Tomcat Exporter نیز به‌صورت پیش‌فرض از طریق یک آدرس HTTP در دسترس قرار می‌گیرند و معمولاً فاقد سازوکارهای داخلی احراز هویت و رمزنگاری هستند. بنابراین لازم است دسترسی به آن‌ها با استفاده از تنظیمات شبکه یا سایر لایه‌های امنیتی کنترل شود.

در ادامه مراحل نصب و راه‌اندازی Tomcat Exporter با جزئیات تشریح می‌گردد.

مرحله ۱: دانلود پکیج

همانند پکیج‌های دیگر، فایل نصبی ابزار را باید در سیستم دارای اینترنت دانلود نموده و سپس به سرور منتقل نمود. البته شایان ذکر است در حال حاضر، پروژه Tomcat Exporter به‌طور فعال توسعه داده نمی‌شود، زیرا قابلیت‌های آن توسط ابزار عمومی‌تر JMX Exporter پوشش داده شده و جامعه‌ی توسعه، تمرکز خود را بر این ابزار جایگزین قرار داده است. با این وجود، برای دانلود Tomcat Exporter می‌توان از لینک زیر استفاده نمود:

https://repo1.maven.org/maven2/nl/nlighten/tomcat_exporter_servlet/0.0.15/

مرحله ۲: انتقال فایل دانلود شده به سرور

فایل دانلود شده در مرحله قبل، همانند مرحله ۶: بخش پیشین، بر روی سرور پایش‌شونده و داخل پوشه webapps از فولدر tomcat9، بارگذاری می‌گردد که پیشنهاد می‌گردد پیش از بارگذاری، نام فایل، به نامی ساده‌تر تغییر یابد.

مرحله ۳: دانلود پیش‌نیازها و بارگذاری آن‌ها

با توجه به اینکه فایل WAR دانلودشده شامل همه وابستگی‌های موردنیاز نیست.، لازم است تا فایل‌های وابسته JAR این ابزار بصورت دستی دانلود شده و در مسیر ~/Downloads/tomcat9/webapps/tomcat_exporter/WEB-INF/lib بارگذاری گردند

(لینک دانلود این بسته‌ها در ادامه آورده شده است). اگر پوشه lib در مسیر فوق وجود نداشته باشد، بایستی پوشه با دستور mkdir ساخته شود:

```
cd ~/Downloads/tomcat9/webapps/tomcat_exporter/WEB-INF      # ورود به پوشه هدف
sudo mkdir -p lib                                           # ساخت پوشه هدف
```

https://repo1.maven.org/maven2/io/prometheus/simpleclient/0.15.0/simpleclient-0.15.0.jar
https://repo1.maven.org/maven2/io/prometheus/simpleclient_common/0.15.0/simpleclient_common-0.15.0.jar
https://repo1.maven.org/maven2/io/prometheus/simpleclient_hotspot/0.15.0/simpleclient_hotspot-0.15.0.jar
https://repo1.maven.org/maven2/io/prometheus/simpleclient_servlet/0.15.0/simpleclient_servlet-0.15.0.jar
https://repo1.maven.org/maven2/io/prometheus/simpleclient_servlet_common/0.15.0/simpleclient_servlet_common-0.15.0.jar
https://repo1.maven.org/maven2/nl/nlighten/tomcat_exporter_client/0.0.9/tomcat_exporter_client-0.0.9.jar

مرحله ۴: راه‌اندازی Tomcat Exporter

بعد از مستقر شدن فایل war مربوطه، باید در یک فایل web.xml اقدام به تعریف servlet مربوط به TomcatMetricsServlet کرد تا تامکت بتواند Servlet مربوط به Tomcat Exporter را اجرا نموده و متریک‌ها را از طریق مسیر /metrics در اختیار Prometheus قرار دهد. برای این فرآیند از دستورات زیر پیروی می‌شود:

```
sudo nano ~/Downloads/tomcat9/webapps/tomcat_exporter/WEB-INF/web.xml      # ساخت فایل web.xml

<?xml version="1.0" encoding="UTF-8"?>                                     # محتوی فایل web.xml
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
        version="3.1">

    <display-name>Tomcat Exporter</display-name>

    <servlet>
        <servlet-name>metrics</servlet-name>
        <servlet-
class>nl.nlighten.prometheus.tomcat.TomcatMetricsServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>metrics</servlet-name>
        <url-pattern>/metrics</url-pattern>
    </servlet-mapping>
</web-app>
ctrl+x , Y(yes)                                                            # ذخیره و بستن فایل فوق
sudo ~/Downloads/tomcat9/bin/shutdown.sh                                   # توقف تامکت
sudo ~/Downloads/tomcat9/bin/startup.sh                                    # راه‌اندازی تامکت
```

در پایان، اگر به آدرس سرور پایش‌شونده http://IP:Port/tomcat_exporter/metrics مراجعه شود، داده‌های گردآوری شده نمایش داده می‌شوند، که Prometheus روی سرور پایش‌کننده همین متریک‌ها را از این آدرس جمع‌آوری خواهد کرد.

فصل چهارم

راه‌اندازی سرور پایش‌کننده

٤-١- پایش معیارهای سیستمی

پس از آماده‌سازی سرورهای پایش‌شونده و نصب و راه‌اندازی سرویس‌ها و اکسپورترها، نوبت به سرور پایش‌کننده می‌رسد. سرور پایش‌کننده نقش مرکزی در جمع‌آوری، ذخیره‌سازی و مدیریت داده‌های پایش را بر عهده دارد. این سرور با جمع‌آوری دوره‌ای اطلاعات از اکسپورترهای نصب‌شده روی سرورهای پایش‌شونده، وضعیت سیستم و سرویس‌ها را به‌صورت مستمر رصد می‌کند و امکان مشاهده وضعیت کلی زیرساخت را در یک نگاه فراهم می‌آورد. تمامی متریک‌های جمع‌آوری‌شده در این سرور ذخیره شده و مبنای تهیه گزارش‌ها، ترسیم نمودارها، تعریف هشدارها و تحلیل روند عملکرد سامانه قرار می‌گیرند. ازاین‌رو، عملکرد صحیح سرور پایش‌کننده نقش مستقیمی در کیفیت و دقت فرآیند پایش کل زیرساخت خواهد داشت.

در این مرحله، نصب و پیکربندی ابزارهای پایش مانند Prometheus یا سایر سیستم‌های پایش مرکزی اهمیت ویژه‌ای دارد. سرور پایش‌کننده باید توانایی برقراری ارتباط پایدار با تمام سرورهای هدف و اکسپورترهای فعال را داشته باشد و داده‌ها را در بازه‌های زمانی مشخص و با حداقل تاخیر جمع‌آوری کند. علاوه بر این، امکان تعریف قواعد پایش، ثبت هشدارها و یکپارچه‌سازی با ابزارهای نمایش داده، مانند Grafana، باعث می‌شود وضعیت منابع و سرویس‌ها به‌صورت متمرکز قابل مشاهده و تحلیل باشد. این قابلیت‌ها به مدیر سامانه کمک می‌کنند تا تغییرات غیرعادی را در کوتاه‌ترین زمان شناسایی کرده و پیش از ایجاد اختلال گسترده، اقدامات لازم را انجام دهد.

به‌طور کلی، سرور پایش‌کننده علاوه بر ذخیره‌سازی متریک‌ها، امکان تحلیل روند تغییرات عملکرد سیستم را نیز فراهم می‌کند. اطلاعات ذخیره‌شده در طول زمان، تصویری از رفتار سامانه در شرایط مختلف کاری ارائه می‌دهد و امکان مقایسه عملکرد در بازه‌های زمانی گوناگون را فراهم می‌سازد. این اطلاعات مبنایی برای ارزیابی سلامت زیرساخت، برنامه‌ریزی ظرفیت، شناسایی گلوگاه‌های عملکردی و بهینه‌سازی سرویس‌ها محسوب می‌شود. همچنین در زمان بروز خطا، داده‌های ثبت‌شده می‌توانند در ریشه‌یابی مشکلات و تشخیص علت اختلال نقش مؤثری ایفا کنند. به همین دلیل، وجود یک سرور پایش‌کننده مناسب یکی از اجزای اصلی زیرساخت‌های پایش مدرن به شمار می‌رود و نقش مهمی در حفظ پایداری و قابلیت اطمینان سرویس‌ها دارد.

۴-۱-۱- نصب Prometheus

برای شروع پایش مرکزی، نخستین گام، نصب و پیکربندی Prometheus بر روی سرور پایش‌کننده است. Prometheus یک سامانه‌ی پایش متن‌باز است که برای جمع‌آوری و ذخیره‌سازی داده‌های زمان‌محور[۷۸] طراحی شده و از پروتکل HTTP برای جمع‌آوری داده‌ها از اکسپورترهای فعال سرورهای پایش‌شونده استفاده می‌کند. نصب Prometheus شامل دریافت نسخه رسمی، ایجاد ساختار

[۷۸] time-series

مناسب برای فایل‌های پیکربندی و داده‌ها، و راه‌اندازی سرویس با استفاده از systemd[۷۹] یا سایر روش‌های متدوال مدیریت سرویس است.

پس از نصب، مرحله‌ی پیکربندی اهمیت ویژه‌ای دارد. در فایل پیکربندی بایستی سرورهای هدف و اکسپورترهای فعال را تعریف نمود تا Prometheus بتواند متریک‌ها را به‌صورت منظم جمع‌آوری کند. همچنین تنظیمات مربوط به دوره‌ی نگهداری داده‌ها[۸۰] و قواعد[۸۱] ثبت شده برای هشداردهی[۸۲] در این فایل انجام می‌شوند. پس از اعمال پیکربندی و راه‌اندازی سرویس، می‌توان از طریق رابط وب Prometheus از صحت عملکرد سرویس و برقراری ارتباط با سرورهای پایش‌شونده اطمینان حاصل کرد. در ادامه مراحل نصب و راه‌اندازی این ابزار گام به گام تشریح می گردد:

مرحله ۱: درخواست دسترسی به سرور پایش‌کننده

به‌منظور دسترسی به مدیریت سرور پایش‌کننده، درخواست دسترسی خواندن/ نوشتن[۸۳] به سرور هدف از واحد فناوری اطلاعات سازمان درخواست می‌گردد.

مرحله ۲: ورود به سرور پایش‌کننده

همانند مرحله ۲: فصل قبل، گام بعدی ورود به محیط اسکریپت‌نویسی لینوکسی سرور هدف است.

مرحله ۳: دانلود Prometheus

با توجه به اینکه اغلب سرورهای سازمانی محدودیت دسترسی به اینترنت آزاد دارند، بایستی پکیج ابزار در نسخه متناسب با سیستم عامل لینوکس، در سیستمی که دسترسی به اینترنت دارد (رایانه شخصی) دانلود شده و فایل دانلود شده به سرور و داخل پوشه مدنظر (پوشه Downloads/payesh) منتقل گردد. برای دانلود به آدرس زیر می‌توان مراجعه نمود:

https://github.com/prometheus/prometheus/releases/download/v2.54.1/prometheus-2.54.1.linux-amd64.tar.gz

مرحله ۴: انتقال فایل دانلود شده به سرور

به‌این منظور یک پوشه به‌نام payesh ساخته و فایل دانلود شده درون این پوشه بارگذاری می‌شود:

```
sudo mkdir ~/payesh                    # ساخت پوشه هدف
```

---

[۷۹] سامانه‌ی مدیریت سرویس و راه‌اندازی در بسیاری از توزیع‌های لینوکس است که مسئول اجرای خودکار، توقف، نظارت و مدیریت سرویس‌ها در سطح سیستم‌عامل می‌باشد.

[۸۰] Retention: مشخص می‌کند متریک‌های ذخیره‌شده چه مدت در پایگاه داده باقی مانده و پس از گذشت این زمان به‌منظور مدیریت حجم ذخیره‌سازی به‌طور خودکار حذف شوند.

[۸۱] Rules

[۸۲] Alert

[۸۳] Read & Write

مرحله ۵: نصب Prometheus

پس از اینکه فایل در مسیر مدنظر روی سرور قرار گرفت، وارد پوشه مربوطه شده و فایل دانلود شده را از حالت فشرده خارج نموده که محتوای آن استخراج می‌شود و بهتر است جهت سهولت در فراخوانی‌های آتی نام ساده‌تری به آن اختصاص یابد:

```
cd ~/payesh                                                    # ورود به پوشه حاوی فایل دانلود شده prometheus
tar -xvf prometheus-2.54.1.linux-amd64.tar.gz                  # خارج کردن از حالت فشرده و استخراج محتوا
sudo mv prometheus-2.54.1.linux-amd64 prometheus               # تغییرنام پوشه به prometheus
```

مرحله ۶: ساخت پوشه داده

Prometheus داده‌ها را به صورت فایل‌های پایگاه داده‌ی سری زمانی محلی[۸۴] ذخیره می‌کند، لذا لازم است پوشه به نام data داخل مسیر prometheus ساخته و سپس هنگام اجرای Prometheus پارامتر مربوطه را تعریف نمود.

```
mkdir ~/payesh/prometheus/data                                 # ساخت پوشه داده جهت ذخیره‌سازی سری زمانی
```

مرحله ۷: ویرایش فایل پیکربندی

در این مرحله بایستی اقدام به ویرایش فایل پیکربندی نموده که در واقع قلب تنظیمات Prometheus است. به این منظور:

```
sudo nano ~/payesh/prometheus/prometheus.yml                   # ورود به محیط ویرایش فایل
```

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds.
Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The
default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the
global 'evaluation_interval'.
rule_files:
  - /home/user/payesh/prometheus/rules/*.yml
  # - "second_rules.yml"
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any
timeseries scraped from this config.
```

scrape_interval:
فاصله‌ی زمانی‌ای که Prometheus سراغ هر target رفته و داده‌های آن را می‌خواند (هر ۱۵ ثانیه)
evaluation_interval: فاصله‌ی اجرای rule‌ها

Alertmanager:
سرویسی که وظیفه ارسال نوتیفیکیشن‌ها مانند ایمیل/تلگرام را دارد و فعلاً خالی است

قوانین از آدرس فایل‌های rule فراخوانی می‌شوند.

scrape_configs:
در این job، Prometheus خودش را (که روی localhost:9090 اجرا می‌شود) scrape می‌کند.

[۸۴] TSDB (Time Series Database)

```
- job_name: "prometheus"
  # metrics_path defaults to '/metrics'
  # scheme defaults to 'http'.
  static_configs:
    - targets: ["IP:9090"]
- job_name: "server1"
  static_configs:
    - targets: ["IP:9100"]
- job_name: "server1_tomcat"
  metrics_path: "/tomcat_exporter"
  static_configs:
    - targets: ["IP:8080"]
```

یعنی خود Prometheus هم یک target هست و metrics داخلی خودش را می‌دهد.
جاب‌های دیگر مربوط به اکسپورترها هستند.

پورت ۹۱۰۰ مربوط به node_exporter هست که در فصل قبل تعریف شد.

پورت ۸۰۸۰ مربوط به tomcat_exporter هست که در فصل قبل تعریف شد.

مرحله ۸: ایجاد قوانین ذخیره لاگ قطعی سرور و تامکت

متریک up در Prometheus وضعیت دسترس‌پذیری هر هدف پایش را در هر نوبتِ جمع‌آوری ثبت می‌کند و مشخص می‌سازد که در آن لحظه ارتباط با هدف پایش برقرار بوده یا خیر. هرچند تاریخچه این متریک در پایگاه داده Prometheus ذخیره می‌شود، اما برای ثبت صریح زمان وقوع رخدادهای قطع و وصل و استفاده ساده‌تر از این اطلاعات در گزارش‌ها، نمودارها و محاسبات بعدی، تعریف Recording Rule مناسب ضروری است. به این منظور، ابتدا پوشه‌ای برای نگهداری فایل‌های Rule ایجاد کرده و سپس فایل Rule زیر ساخته می‌شود:

```
sudo mkdir -p /payesh/prometheus/rules
sudo tee /home/user/payesh/prometheus/rules/server_downtime.yml > /dev/null <<'EOF'
groups:
- name: server_downtime_rules
  interval: 15s
  rules:

    - record: server_down_event
      expr: (up{job="server1"} == 0)
            and on(job, instance)
            (up{job="server1"} offset 15s == 1)

    - record: server_up_event
      expr: (up{job="server1"} == 1)
            and on(job, instance)
            (up{job="server1"} offset 15s == 0)

EOF
```

ساخت پوشه حاوی rule#
ساخت فایل rule#
بجای user نام کاربری وارد شود #

در اقدامی مشابه یک قاعده برای ذخیره مقاطع قطعی تامکت در زمان‌هایی که سرور روشن است، تعریف می‌شود:

```
cat > /home/user/payesh/prometheus/tomcat_downtime_rules.yml <<'EOF'
groups:
  - name: tomcat_downtime_rules
    interval: 15s
```

```
  rules:
    - record: server_up_by_host
      expr: label_replace(up{job="server1"}, "host", "$1", "instance", "([^:]+):.*")

    - record: tomcat_up_by_host
      expr: label_replace(up{job="server1_tomcat"}, "host", "$1", "instance", "([^:]+):.*")

    - record: tomcat_down_event
      expr: (tomcat_up_by_host == 0) and on(host) (tomcat_up_by_host offset 15s == 1) and
on(host) (server_up_by_host == 1) and on(host) (server_up_by_host offset 15s == 1) and
on(host) (server_up_by_host offset 30s == 1)

    - record: tomcat_up_event
      expr: (tomcat_up_by_host == 1) and on(host) (tomcat_up_by_host offset 15s == 0) and
on(host) (server_up_by_host == 1) and on(host) (server_up_by_host offset 15s == 1) and
on(host) (server_up_by_host offset 30s == 1) and on(host) (server_up_by_host offset 45s == 1)
and on(host) (server_up_by_host offset 60 == 1)

    - record: tomcat_down_time
      expr: time() * tomcat_down_event

    - record: tomcat_up_time
      expr: time() * tomcat_up_event
EOF
```

مرحله ۹: اجرای دستی Prometheus

پس از تکمیل فایل پیکربندی، می‌توان Prometheus را برای بررسی اولیه و اطمینان از صحت راه‌اندازی، به‌صورت دستی اجرا کرد. برای این منظور، ابتدا وارد پوشه نصب Prometheus شده و سپس فایل اجرایی آن همراه با مسیر فایل پیکربندی و پوشه ذخیره‌سازی داده‌ها اجرا شود:

```
cd payesh/prometheus                                                    # ورود به پوشه
./prometheus --config.file=prometheus.yml --storage.tsdb.path=data      # اجرای دستی
```

مرحله ۱۰: خودکارسازی راه‌اندازی Prometheus

برای اینکه Prometheus پس از هر بار راه‌اندازی مجدد سرور به‌صورت خودکار اجرا شود و نیازی به اجرای دستی آن نباشد، یک فایل سرویس برای systemd ایجاد نموده و پس از ذخیره فایل، پیکربندی systemd بازخوانی شده و سرویس Prometheus فعال و راه‌اندازی می‌شود. در این حالت، اجرای مرحله قبل صرفاً برای آزمایش دستی مورد نیاز خواهد بود.

```
sudo tee /etc/systemd/system/prometheus.service > /dev/null << 'EOF'      # ایجاد فایل سرویس
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target
[Service]
User=user
Group=user
Type=simple
ExecStart=/home/user/payesh/prometheus/prometheus \
  --config.file=/home/user/payesh/prometheus/prometheus.yml \
  --storage.tsdb.path=/home/user/payesh/prometheus/data \
  --query.max-samples=2000000
Restart=always
[Install]
WantedBy=multi-user.target
EOF

sudo systemctl daemon-reload                                              # بازخوانی فایل systemd
sudo systemctl enable prometheus                                          # فعال‌سازی سرویس Prometheus
sudo systemctl start prometheus                                           # استارت سرویس Prometheus
sudo systemctl status prometheus -l                                       # وضعیت سرویس Prometheus
```

۴-۱-۲- داشبورد Prometheus

پس از نصب و راه‌اندازی موفق Prometheus، نخستین نقطه تعامل کاربر با این سامانه، رابط وب داخلی آن است که به‌صورت پیش‌فرض بر روی پورت ۹۰۹۰ فعال می‌شود. این رابط وب در آدرس زیر در دسترس خواهد بود:

http://<IP سرور : 9090>

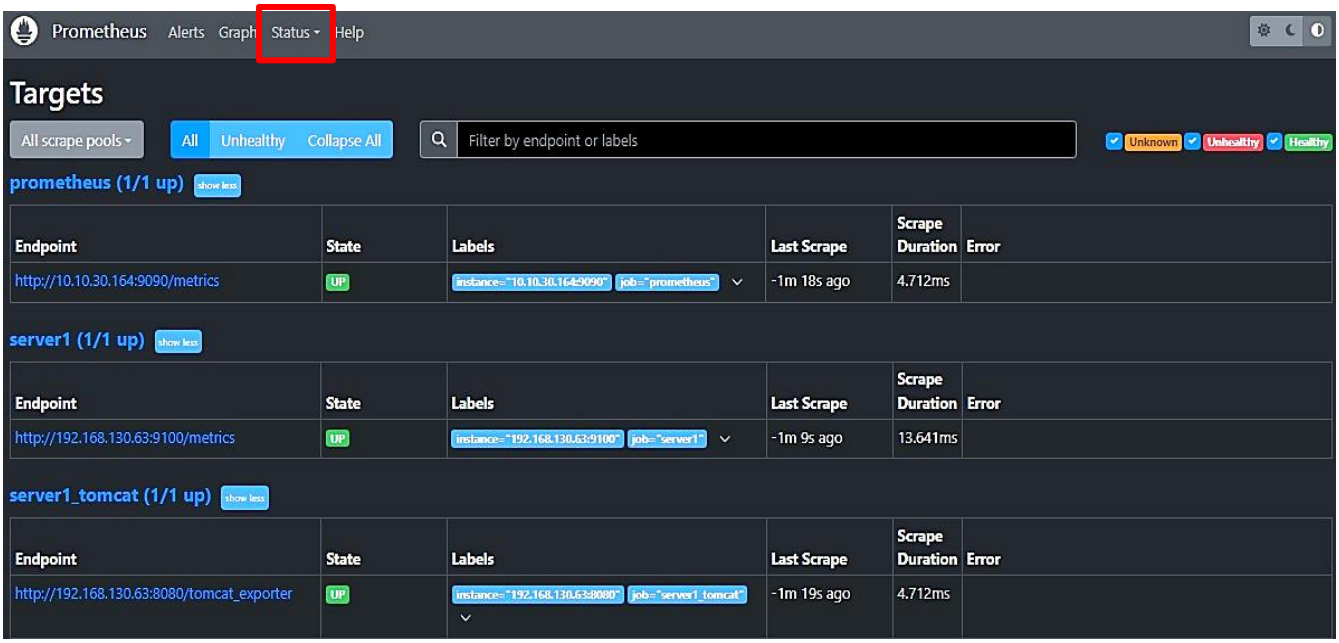
رابط وب Prometheus محیطی سبک و تحت وب برای مدیریت و بررسی وضعیت سامانه پایش فراهم می‌کند و بدون نیاز به نصب نرم‌افزار جانبی، تنها از طریق مرورگر قابل استفاده است. از طریق این رابط می‌توان وضعیت اهداف پایش، صحت عملکرد اکسپورترها، نتایج پرس‌وجوهای PromQL، متریک‌های جمع‌آوری‌شده و قواعد تعریف‌شده را مشاهده و بررسی کرد. این محیط علاوه بر عیب‌یابی اولیه، امکان اطمینان از صحت عملکرد Prometheus و ارتباط آن با سرورهای پایش‌شونده را نیز فراهم می‌سازد. مهم‌ترین بخش‌های این رابط وب عبارت‌اند از:

۱. Status → Targets

در این بخش، تمامی اهداف[۸۵] پایش که در فایل پیکربندی تعریف شده‌اند نمایش داده می‌شوند. هر تارگت نشان‌دهنده یک منبع جمع‌آوری متریک، مانند خود Prometheus، یک Node Exporter یا یک Tomcat Exporter است. وضعیت هر تارگت

[۸۵] Targets

به‌صورت برخط قابل مشاهده است و در هر نوبت جمع‌آوری به‌روزرسانی می‌شود. نمایش وضعیت سبز (Up) بیانگر موفقیت در جمع‌آوری داده‌ها و متریک‌ها بوده و وضعیت قرمز (Down) نشان‌دهنده وجود مشکل در ارتباط شبکه، پیکربندی، یا در دسترس نبودن سرویس هدف است. علاوه بر وضعیت، اطلاعاتی مانند زمان آخرین جمع‌آوری، مدت زمان جمع‌آوری متریک‌ها و آدرس تارگت نیز در این بخش نمایش داده می‌شود که در عیب‌یابی و بررسی صحت عملکرد سامانه بسیار مفید هستند.



شکل ٤-١- نمایی از وضعیت تارگت‌ها در Prometheus

۲. Graph

این بخش امکان اجرای پرس‌وجوهای PromQL را فراهم می‌کند. کاربر می‌تواند متریک‌های مختلف، مانند مصرف پردازنده، حافظه، دیسک یا شبکه را انتخاب کرده و نتایج آن‌ها را به‌صورت جدول یا نمودار زمانی مشاهده کند. این قابلیت برای بررسی وضعیت فعلی سامانه، مقایسه متریک‌ها و تحلیل روند تغییرات آن‌ها در طول زمان بسیار کاربردی است.

مطابق شکل ۴-۲، برای واردسازی پرس‌وجو، پس از انتخاب منوی گراف، در کادر Expression عبارت PromQL را وارد کرده و پس از اجرا[۸۶] (کادر شماره ۱)، نتایج هم به‌صورت جدولی و هم به‌صورت نموداری (کادر شماره ۲) ظاهر می‌شوند. در خروجی نتایج instance و job (کادر شماره ۳) مشخص می‌کنند که کدام تارگت رصد می‌شود. البته می‌توان در Expression سرور تارگت را تعیین نمود.

[۸۶] Execute



شکل ٤-۲ – نمونه‌ای از نمودار پرس‌وجو در گراف Prometheus

ازجمله پرکاربردترین پرس‌وجوها عبارتند از:

- پایش پردازنده:

```
node_cpu_seconds_total                                  # کل زمان CPU به تفکیک (user/system/idle)
node_cpu_seconds_total{instance="IP:9100"}              # پایش CPU فقط روی سرور۱
node_cpu_seconds_total{mode="idle"}                     # فقط پردازنده در حالت idle
 100- (avg by(instance)                                 # درصد استفاده واقعی از CPU (میانگین یک دقیقه)
(irate(node_cpu_seconds_total{mode="idle"}[1m])) * 100)
```

- پایش حافظه:

```
node_memory_MemTotal_bytes                              # کل حافظه سیستم
node_memory_MemAvailable_bytes                          # حافظه در دسترس
(1 - (node_memory_MemAvailable_bytes /                  # درصد استفاده از حافظه
node_memory_MemTotal_bytes)) * 100
```

- پایش دیسک:

```
node_filesystem_size_bytes                              # فضای کل دیسک (به ازای هر mountpoint)
node_filesystem_free_bytes                              # فضای آزاد دیسک
(1 - (node_filesystem_free_bytes /                      # درصد استفاده از دیسک
node_filesystem_size_bytes)) * 100
```

- پایش شبکه:

```
rate(node_network_receive_bytes_total[1m])              # بایت‌های دریافتی روی کارت شبکه
rate(node_network_transmit_bytes_total[1m])             # بایت‌های ارسال‌شده روی کارت شبکه
rate(node_network_receive_errs_total[1m])               # خطاهای کارت شبکه
```

۳. Alerts

در صورت تعریف قوانین هشدار[۸۷]، وضعیت هشدارها در این بخش نمایش داده می‌شود. به کمک آن می‌توان رویدادهای بحرانی مانند پرشدن دیسک یا مصرف بیش‌ازحد منابع را شناسایی و مدیریت کرد. در صورتی که هیچ قانونی تعریف نشده باشد، این بخش خالی خواهد بود.

برای تعریف قانون هشدار می‌توان یک فایل rule با فرمت yml. ساخته و در پوشه payesh/prometheus گذاشته و در فایل پیکربندی prometheus.yml در قسمت قوانین، فایل مربوطه را فراخوانی نمود. به‌عنوان مثال:

```
sudo nano payesh/prometheus/alerts.yml                    # ساخت فایل rule هشدار
groups:                                                   # محتوی فایل rule
  - name: example
    rules:
      - alert: HighCPUUsage
        expr: 100 - (avg by(instance)
(irate(node_cpu_seconds_total{mode="idle"}[1m])) * 100) > 80
        for: 1m
        labels:
          severity: warning
        annotations:
          summary: "CPU usage is high on {{ $labels.instance }}"
          description: "CPU utilization is above 80% for more than 1 minute."
ctrl+x , Y (yes)                                          # ذخیره‌سازی و بستن فایل
sudo nano payesh/prometheus/prometheus.yml                # ویرایش فایل پیکربندی
rule_files:                                               # فراخوانی فایل ساخته شده
  - "alerts.yml"
```

در این مثال اگر CPU بیشتر از یک دقیقه در حالت بالای ۸۰٪ باقی بماند، در منوی Alerts وضعیت Firing مشاهده می‌شود.



شکل ٤-۳ – نمونه‌ای از هشدار تعریف‌شده در Prometheus

[۸۷] Alert Rules

به طور کلی، داشبورد وب Prometheus ابزاری ساده ولی کارآمد برای مشاهده و اعتبارسنجی وضعیت پایش است. هرچند قابلیت‌های نمایشی آن محدود بوده و بیشتر بر پرس‌وجوی داده‌ها و بررسی سلامت سیستم تمرکز دارد، اما همین امکانات پایه به کاربر اجازه می‌دهد از صحت عملکرد سامانه اطمینان حاصل کند. برای ترسیم داشبوردهای پیشرفته و گرافیک‌های تحلیلی گسترده‌تر، معمولاً از نرم‌افزارهای دیگر مانند Grafana در کنار Prometheus استفاده می‌شود.

٤-۲- بصری‌سازی پایش

Grafana یک پلتفرم متن‌باز برای پایش، مصورسازی[۸۸] و تحلیل داده‌ها است که با هدف ایجاد داشبوردهای پویا، تعاملی و قابل تنظیم توسعه یافته است. این ابزار امکان نمایش داده‌های جمع‌آوری‌شده از منابع مختلف، از جمله Prometheus، InfluxDB، Elasticsearch و سایر منابع داده را در قالب نمودارها، جداول و سایر اجزای گرافیکی فراهم می‌کند.

Grafana با بهره‌گیری از رابط کاربری تحت وب، امکان طراحی داشبوردهای سفارشی را در اختیار مدیر سامانه قرار می‌دهد تا معیارهای عملکردی، وضعیت سرویس‌ها و روند تغییرات آن‌ها در بازه‌های زمانی مختلف به‌صورت متمرکز قابل مشاهده و تحلیل باشند. همچنین قابلیت بزرگ‌نمایی روی بازه‌های زمانی، مقایسه داده‌ها و اعمال فیلترهای مختلف، تحلیل رخدادها و بررسی تغییرات عملکرد سامانه را ساده‌تر می‌کند. به همین دلیل، Grafana یکی از اجزای اصلی سامانه‌های پایش مدرن محسوب می‌شود و معمولاً در کنار Prometheus برای نمایش و تحلیل متریک‌های جمع‌آوری‌شده به‌کار می‌رود.

از مهم‌ترین ویژگی‌های Grafana می‌توان به موارد زیر اشاره کرد:

- پشتیبانی از چندین منبع داده
- ایجاد و اشتراک‌گذاری داشبوردهای تعاملی
- امکان تعریف هشدارها بر اساس آستانه‌های مشخص
- مدیریت کاربران و نقش‌ها برای کنترل دسترسی
- قابلیت توسعه و یکپارچه‌سازی با سایر سامانه‌ها از طریق افزونه‌ها[۸۹]

در معماری حاضر، Grafana به‌عنوان رابط تصویری[۹۰] سامانه‌ی پایش عمل می‌کند و و متریک‌های ذخیره‌شده در Prometheus را به‌صورت نمودارها، جداول و شاخص‌های قابل تحلیل نمایش می‌دهد. این قابلیت باعث می‌شود وضعیت منابع سخت‌افزاری، سرویس‌ها و روند تغییرات آن‌ها در یک محیط متمرکز و قابل فهم برای مدیر سامانه قابل مشاهده باشد.

---

[۸۸] Visualization

[۸۹] Plugins

[۹۰] Visualization Layer

۴-۲-۱- نصب Grafana

فرآیند نصب Grafana نسبتاً ساده است و می‌تواند به‌صورت آنلاین از مخازن رسمی یا به‌صورت آفلاین از بسته‌های نصب محلی انجام شود. در محیط‌های سازمانی که دسترسی مستقیم به اینترنت وجود ندارد، معمولاً روش آفلاین انتخاب می‌شود تا وابستگی به منابع خارجی حذف شده و کنترل بیشتری بر نسخه‌های نصب‌شده وجود داشته باشد. در مراحل بعد از نصب و راه‌اندازی، می‌توان منبع داده را به Grafana معرفی کرد و داشبوردهای پایش را ایجاد نمود تا متریک‌های جمع‌آوری‌شده از سرورهای پایش‌شونده به‌صورت گرافیکی نمایش داده شوند و امکان تحلیل و پایش مستمر وضعیت سامانه فراهم گردد.

مرحله ۱: دانلود Grafana

این ابزار دو بسته OSS و Enterprise دارد. نسخه OSS رایگان و متن‌باز است و شامل اکثر ویژگی‌های پایه‌ای و حیاتی Grafana برای مصورسازی داده‌ها، ساخت داشبورد، پشتیبانی از منابع داده‌ای مانند Prometheus و ایجاد هشدارهای ساده می‌باشد. اما محدودیت‌هایی در امکانات سازمانی پیشرفته مانند همگام‌سازی سازمانی، مدیریت پیشرفته دسترسی کاربران و پشتیبانی اختصاصی دارد. نسخه Enterprise تمامی امکانات نسخه OSS را داراست و علاوه بر آن ویژگی‌های سازمانی و پیشرفته‌تری ارائه می‌دهد.

از آنجا که امکانات نسخه OSS نیازهای این پروژه را به‌طور کامل پوشش داده و تمامی قابلیت‌های موردنیاز را برای جمع‌آوری، نمایش و تحلیل متریک‌ها در اختیار قرار می‌دهد، در این عملیات از نسخه OSS استفاده شده است؛ که از لینک زیر قابل دانلود می‌باشد:

https://dl.grafana.com/oss/release/grafana10.4.19_amd64.deb

مرحله ۲: نصب Grafana

فایل دانلود شده را، درون پوشه payesh بارگذاری نموده و پس از اطمینان از صحت انتقال فایل، بسته نصب Grafana با استفاده از دستور زیر به‌صورت آفلاین نصب می‌شود:

```
# نصب Grafana
sudo dpkg -i grafana_10.4.19_amd64.deb
```

مرحله ۳: تعیین/ تغییر پورت Grafana

پورت پیش‌فرض تخصیص‌یافته به Grafana برابر با ۳۰۰۰ می‌باشد؛ که برای تغییر پورت پیش‌فرض Grafana می‌توان دستورات زیر را وارد نمود:

```
# ویرایش پورت Grafana
sudo nano /etc/grafana/grafana.ini
[server]
# Protocol (http, https, h2, socket)
;protocol = http

# This is the minimum TLS version allowed. By default,
this value is empty. Accepted values are: TLS1.2, TLS1.3.
If nothing is set TLS1.2 would be taken
;min_tls_version = ""

# The ip address to bind to, empty will bind to all interfaces
; در بخش [سرور] پورت را تغییر داده و بایستی علامت
# ابتدایی پاک شود
```

```
;http_addr =

# The http port  to use
http_port = 3001
```

مرحله ۴: تعیین نام کاربری و رمز عبور Grafana

برای تعیین/ تغییر رمز عبور کاربر مدیر (admin) در داشبورد Grafana از دستور زیر استفاده می‌شود:

```
sudo grafana-cli admin reset-admin-password admin        # تعیین/ تغییر رمز عبور
```

پارامتر آخر این دستور، رمز عبور جدید می‌باشد که در این مثال مقدار admin تعریف شده است.

مرحله ۵: راه‌اندازی Grafana

به دلیل اینکه Grafana به‌صورت پیش‌فرض از systemd پشتیبانی می‌کند، نیازی به تعریف فایل سرویس جدید نیست. برای راه‌اندازی آن دستورات زیر وارد می‌شود:

```
sudo systemctl daemon-reload          # بازخوانی پیکربندی systemd
sudo systemctl enable grafana-server  # فعالسازی Grafana
sudo systemctl start grafana-server   # استارت Grafana
sudo systemctl status grafana-server  # وضعیت Grafana
```

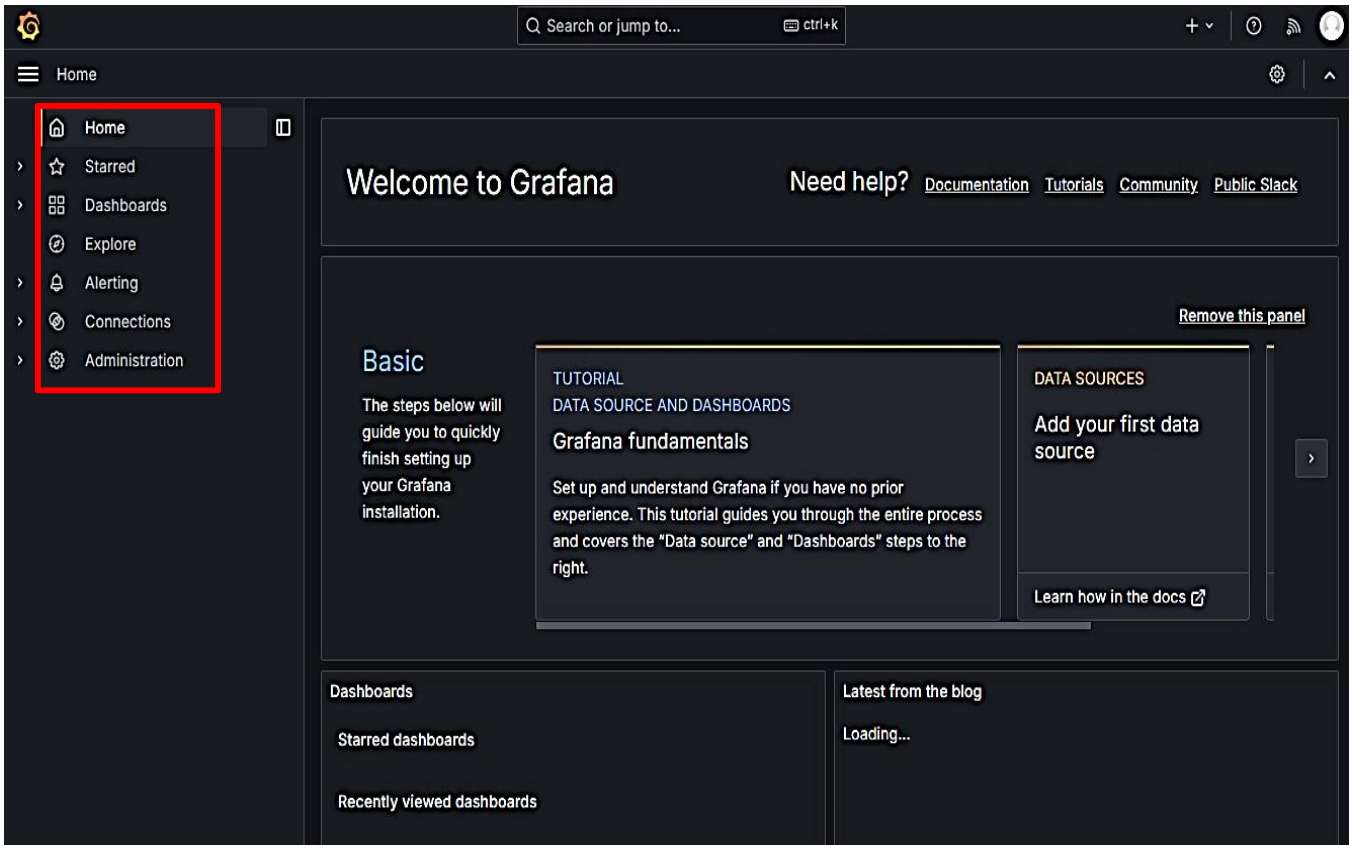
۴-۲-۲- داشبورد Grafana

پس از نصب و راه‌اندازی موفق Grafana، محیط تحت وب آن از طریق مرورگر و در پورت پیش‌فرض ۳۰۰۰ یا پورت تنظیم‌شده در فایل grafana.ini (در مثال این مستند ۳۰۰۱) در دسترس خواهد بود. رابط کاربری Grafana یک محیط گرافیکی مدرن، پویا و کاملاً تعاملی است که امکان نمایش و تحلیل متریک‌های جمع‌آوری‌شده از منابع داده مختلف را در قالب نمودارها، جداول، شاخص‌های عملکردی[۹۱] و سایر مؤلفه‌های بصری فراهم می‌کند. علاوه بر این، کاربر می‌تواند بازه زمانی نمایش داده‌ها را تغییر داده، داشبوردها را سفارشی‌سازی کرده و وضعیت سامانه را به‌صورت برخط پایش کند.

در اولین ورود، کاربر با استفاده از نام کاربری و رمز عبور تعریف شده در مرحله ۴: نصب، وارد سامانه شده و در صورت استفاده از رمز عبور پیش‌فرض، Grafana درخواست تغییر آن را نمایش می‌دهد که کاربر می‌تواند در این گام رمز عبور جدیدی تعیین کند.

پس از ورود، صفحه‌ی اصلی شامل منوهایی برای مدیریت منابع داده، ایجاد داشبورد جدید و مدیریت داشبوردها، تعریف و تنظیم هشدارها، نصب افزونه‌ها و مدیریت کاربران و سطوح دسترسی است.

[۹۱] KPI: Key Performance Indicator



شکل ٤-٤ – نمایی از داشبورد Grafana

در ادامه توضیحات مختصری از **منوهایی** که در داشبورد Grafana مطابق شکل ۴-۴ مشاهده می‌شوند، آورده شده است.

۱. Home

صفحه Home نقطه آغاز کار در Grafana است و نمایی کلی از محیط کاربری، همراه با میانبرهایی برای دسترسی سریع به داشبوردها، منابع داده، مستندات و سایر قابلیت‌های سامانه در اختیار کاربر قرار می‌دهد. همچنین از این بخش می‌توان داشبورد جدیدی ایجاد کرد یا به داشبوردهایی که اخیراً مورد استفاده قرار گرفته‌اند، دسترسی یافت.

۲. Starred

بخش Starred محل نمایش داشبوردهایی است که کاربر آن‌ها را به‌عنوان "مورد علاقه" علامت‌گذاری کرده است. این قابلیت کمک می‌کند تا در میان تعداد زیادی داشبورد، دسترسی به موارد مهم یا پرکاربرد سریع‌تر انجام شود. با انتخاب نماد ستاره در بالای هر داشبورد، آن داشبورد به فهرست Starred افزوده شده و از این بخش به‌طور مستقیم قابل دسترسی خواهد بود.

۳. Dashboards

در منوی Dashboards، تمامی داشبوردهای موجود در سامانه فهرست می‌شوند. از این بخش می‌توان داشبورد جدید ایجاد کرد، داشبوردهای موجود را ویرایش یا حذف نمود، و آن‌ها را در پوشه‌های جداگانه برای سازمان‌دهی بهتر دسته‌بندی کرد. هر داشبورد می‌تواند شامل چندین پنل باشد و هر پنل نیز به‌صورت مستقل برای نمایش یک نمودار، جدول، شاخص یا نوع دیگری از نمایش داده

پیکربندی شود. همچنین هر پنل می‌تواند پرس‌وجو، منبع داده، بازه زمانی و تنظیمات نمایشی مخصوص خود را داشته باشد. این انعطاف‌پذیری امکان طراحی داشبوردهای متناسب با نیازهای مختلف مدیریتی، عملیاتی و فنی را فراهم می‌سازد.

۴. Explore

بخش Explore محیطی برای اجرای مستقیم و موقت پرس‌وجوها و بررسی داده‌ها بدون نیاز به ایجاد یا ذخیره داشبورد است. در این قسمت، کاربر می‌تواند یک منبع داده مانند Prometheus را انتخاب کرده، عبارت پرس‌وجو را وارد کند و نتیجه را به‌صورت نمودار یا جدول مشاهده نماید. همچنین امکان مقایسه نتایج پرس‌وجوها، تغییر بازه زمانی و بررسی جزئیات متریک‌ها وجود دارد. این بخش معمولاً برای آزمایش اولیه پرس‌وجوها، تحلیل سریع رخدادها و یافتن عبارت مناسب پیش از استفاده از آن در پنل‌های داشبورد به کار می‌رود.

۵. Alerting

در بخش Alerting، امکان تعریف و مدیریت قواعد هشدار فراهم شده است. کاربر می‌تواند شرایطی مانند افزایش مصرف پردازنده از یک آستانه مشخص یا قطع ارتباط با یک هدف پایش را در قالب یک قاعده هشدار تعریف کند. Grafana این قواعد را در بازه‌های زمانی تعیین‌شده ارزیابی می‌کند و در صورت برقرار شدن شرایط تعریف‌شده، وضعیت هشدار را فعال می‌سازد. چنانچه نقاط تماس و سیاست‌های ارسال اعلان نیز پیکربندی شده باشند، اعلان مربوط به هشدار می‌تواند از طریق روش‌هایی مانند ایمیل، پیام‌رسان یا سایر کانال‌های تعریف‌شده ارسال شود. این بخش برای شناسایی خودکار شرایط غیرعادی و اطلاع‌رسانی به‌موقع به تیم بهره‌برداری استفاده می‌شود.

۶. Connections

منوی Connections برای تعریف و مدیریت ارتباط Grafana با منابع داده و افزونه‌ها[۹۲] استفاده می‌شود. در این بخش، کاربر می‌تواند منابع داده‌ای مانند Prometheus، InfluxDB، MySQL یا Elasticsearch را اضافه و پیکربندی کند. همچنین امکان بررسی صحت اتصال، ویرایش تنظیمات منبع داده و مدیریت افزونه‌های Grafana از این بخش فراهم است. تعریف صحیح منابع داده در این قسمت، پیش‌نیاز اجرای پرس‌وجوها و نمایش اطلاعات در داشبوردها محسوب می‌شود.

۷. Administration

بخش Administration امکانات مدیریتی Grafana را در اختیار کاربران دارای مجوزهای لازم قرار می‌دهد. از این بخش می‌توان کاربران، تیم‌ها، نقش‌ها، مجوزهای دسترسی، حساب‌های خدماتی و برخی تنظیمات عمومی و امنیتی سامانه را مدیریت کرد. سطح دسترسی به گزینه‌های این بخش بر اساس نقش و مجوزهای کاربر تعیین می‌شود؛ بنابراین تمامی گزینه‌های مدیریتی برای همه کاربران قابل مشاهده یا استفاده نیستند.

---

[۹۲] Plugins

۴-۲-۳- اتصال Prometheus به Grafana

Grafana به‌تنهایی ابزاری برای نمایش و تحلیل داده‌ها است و برای جمع‌آوری اطلاعات به منابع داده وابسته است. یکی از محبوب‌ترین و قدرتمندترین این منابع، Prometheus است که داده‌های متریک[۹۳] را از سیستم‌ها و سرویس‌ها جمع‌آوری و به‌صورت داده‌های سری زمانی[۹۴] ذخیره می‌کند. در نتیجه، این دو ابزار مکمل یکدیگرند:

- Prometheus متریک‌ها را از Exporterها (مانند Node Exporter و Tomcat Exporter) جمع‌آوری می‌کند.
- Grafana همین متریک‌ها را از Prometheus می‌خواند و در قالب نمودار، Gauge و Dashboard نمایش می‌دهد.

ارتباط میان Prometheus و Grafana بر پایه معماری «منبع داده[۹۵]» برقرار می‌شود. در این معماری، Grafana هیچ‌گونه متریکی را به‌صورت مستقل جمع‌آوری یا ذخیره نمی‌کند، بلکه هنگام نمایش داشبوردها، پرس‌وجوهای موردنیاز را به Prometheus ارسال کرده و نتایج را دریافت می‌کند. در نتیجه، تمامی نمودارها، جداول و شاخص‌های نمایشی در Grafana بر اساس داده‌هایی هستند که قبلاً توسط Prometheus از Exporterها جمع‌آوری و ذخیره شده‌اند. این تفکیک وظایف موجب افزایش مقیاس‌پذیری، کاهش پیچیدگی سامانه و سهولت توسعه و نگهداری زیرساخت پایش می‌شود.

بدین ترتیب، این اتصال، هسته‌ی اصلی سامانه‌ی پایش ما را شکل می‌دهد؛ یعنی Grafana به Prometheus متصل می‌شود، داده‌ها را از آن واکشی می‌کند، و در قالب داشبوردهای مدیریتی قابل تحلیل نمایش می‌دهد.

به این منظور گام‌های زیر طی می‌شود:

مرحله ۱: اطمینان از صحت عملکرد اکسپورترها و Prometheus

برای بررسی وضعیت اتصال Prometheus به منابع پایش‌شونده، از طریق مرورگر وب و مراجعه به رابط کاربری Prometheus، تمامی تارگت‌های تعریف‌شده در فایل پیکربندی Prometheus قابل مشاهده هستند. هر تارگت نشان‌دهنده یک منبع ارائه‌دهنده متریک، مانند خود Prometheus، Node Exporter یا Tomcat Exporter است که Prometheus به‌صورت دوره‌ای از آن متریک جمع‌آوری می‌کند. قرار داشتن هر تارگت در وضعیت UP به این معناست که ارتباط Prometheus با آن منبع برقرار است و داده‌های لازم برای تحلیل و نمایش در ابزارهایی مانند Grafana قابل جمع‌آوری هستند. در پیکربندی انجام‌شده در این مستند، سه Target اصلی تعریف شده‌اند که عبارت‌اند از:

- خود Prometheus،
- Node Exporter،
- Tomcat Exporter

---

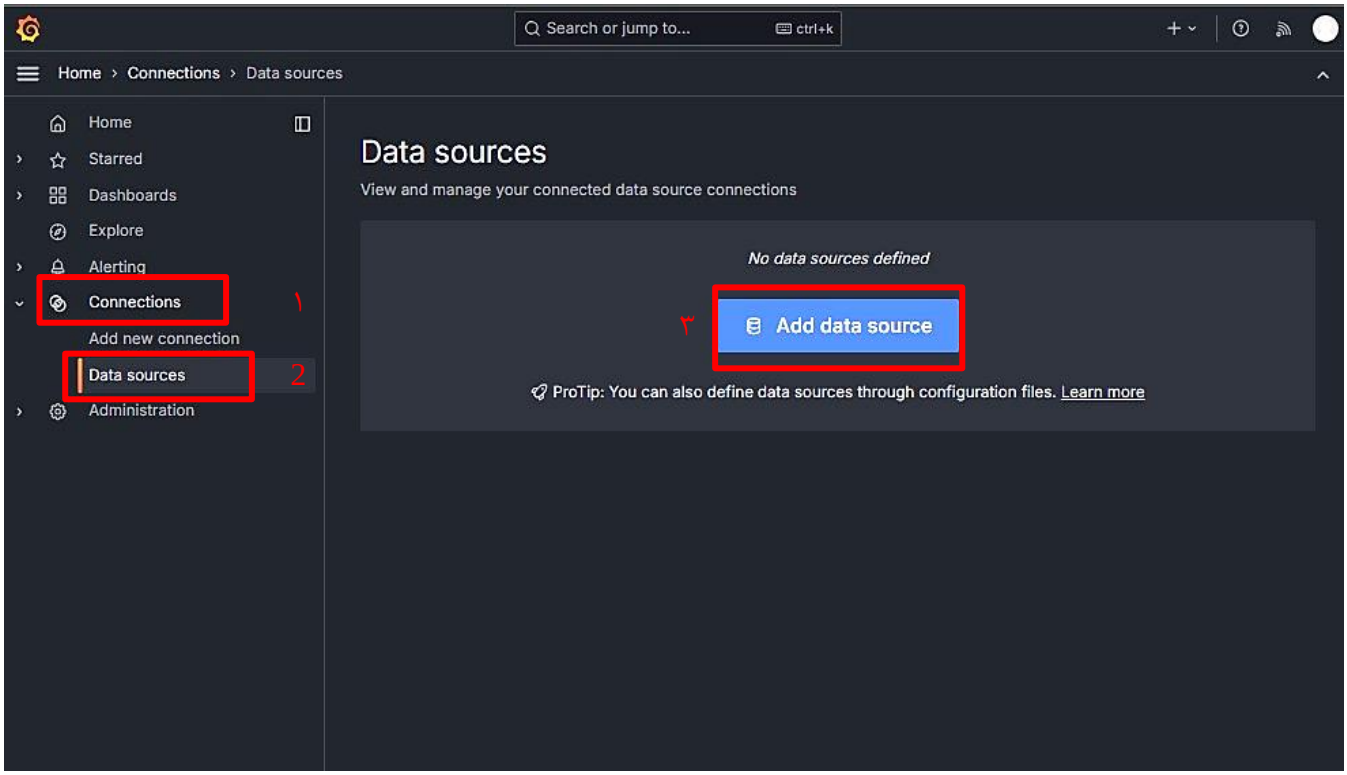
[۹۳] Metrics

[۹۴] Time Series

[۹۵] Data Source

مرحله ۲: ورود به Grafana و اضافه کردن منبع داده

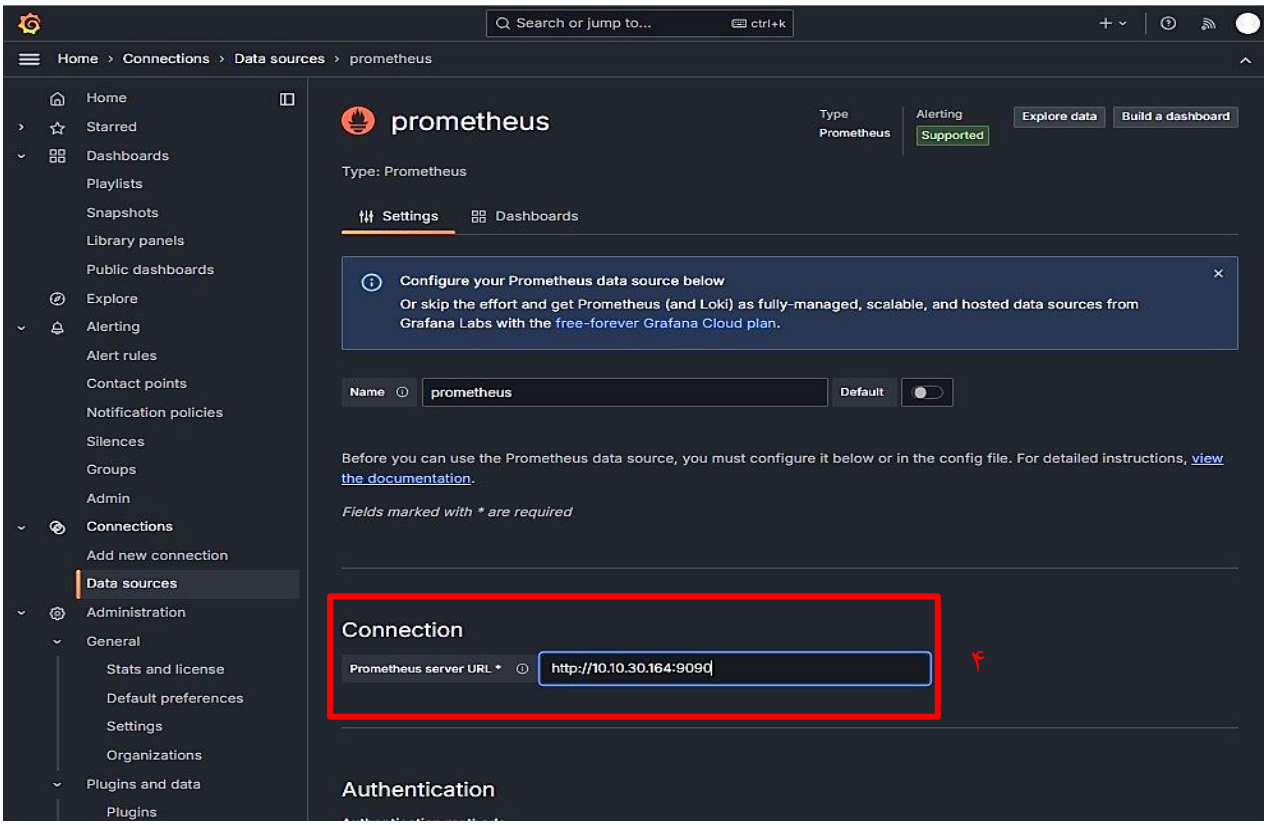
از طریق مرورگر وب، در آدرس Grafana از سرور پایش‌کننده، با حساب کاربری مدیر (admin) وارد شده و مطابق شکل ۴-۵ از منوی سمت چپ، در مسیر Connections > Data Sources و با کلیک بر روی Add data source، پنجره جدید نمایان می‌شود که در آن نوع Data Source، Prometheus انتخاب می‌شود.

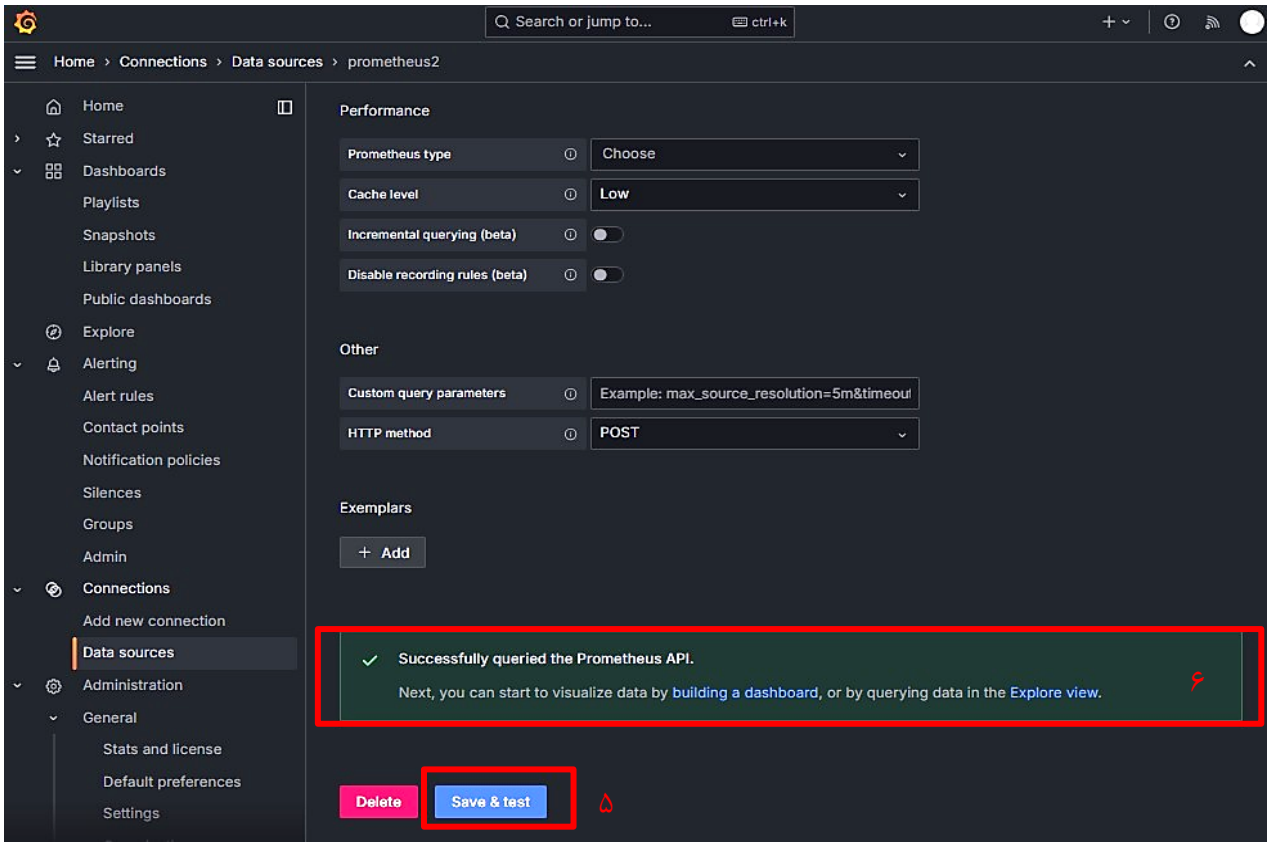


شکل ٤-٥ – مراحل اضافه کردن منابع داده به Grafana

مرحله ۳: تنظیمات اتصال Prometheus

همانند شکل ۴-۶، در بخش URL (کادر شماره ۴)، آدرس سرور Prometheus را وارد نموده و سایر تنظیمات، مانند Access و HTTP Method، را می‌توان به حالت پیش‌فرض باقی گذاشت. سپس در پایین پنجره، با کلیک بر روی دکمه Save & Test (کادر شماره ۵)، اتصال Grafana به Prometheus بررسی می‌شود. در صورت برقراری موفق ارتباط، پیام موفقیت‌آمیز بودن اتصال نمایش داده می‌شود (کادر شماره ۶). از این پس، Grafana می‌تواند متریک‌های ذخیره‌شده در Prometheus را به‌عنوان منبع داده برای ایجاد داشبوردها، اجرای پرس‌وجوها و نمایش نمودارها مورد استفاده قرار دهد.

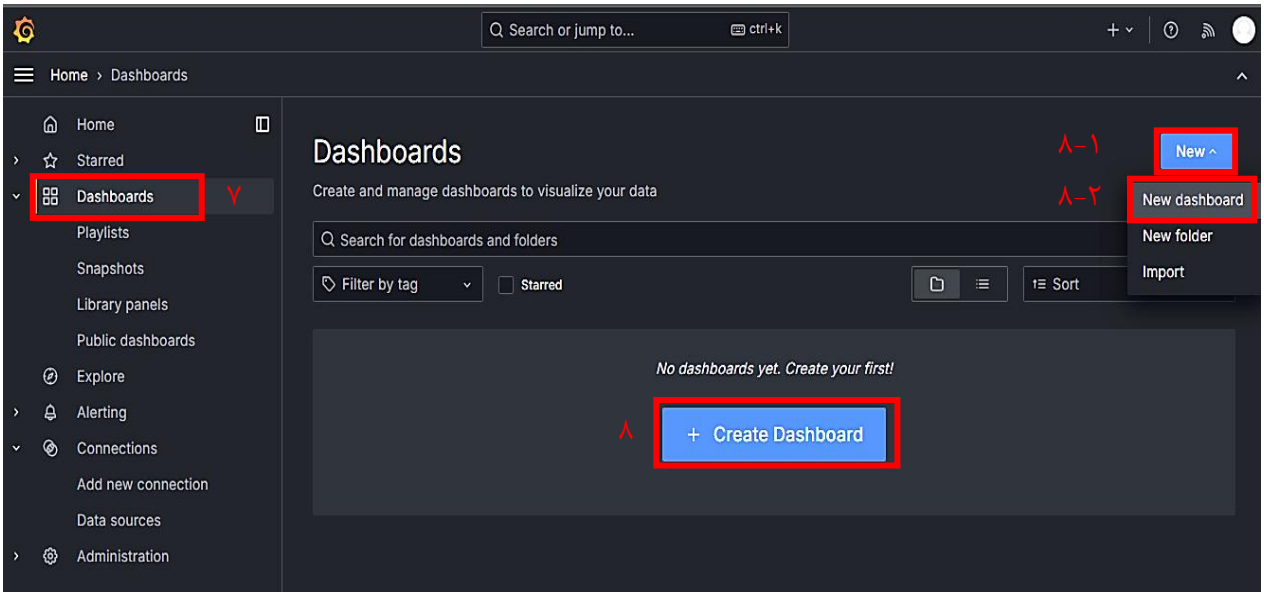




شکل ٤-٦ – تنظیمات اتصال Prometheus به Grafana

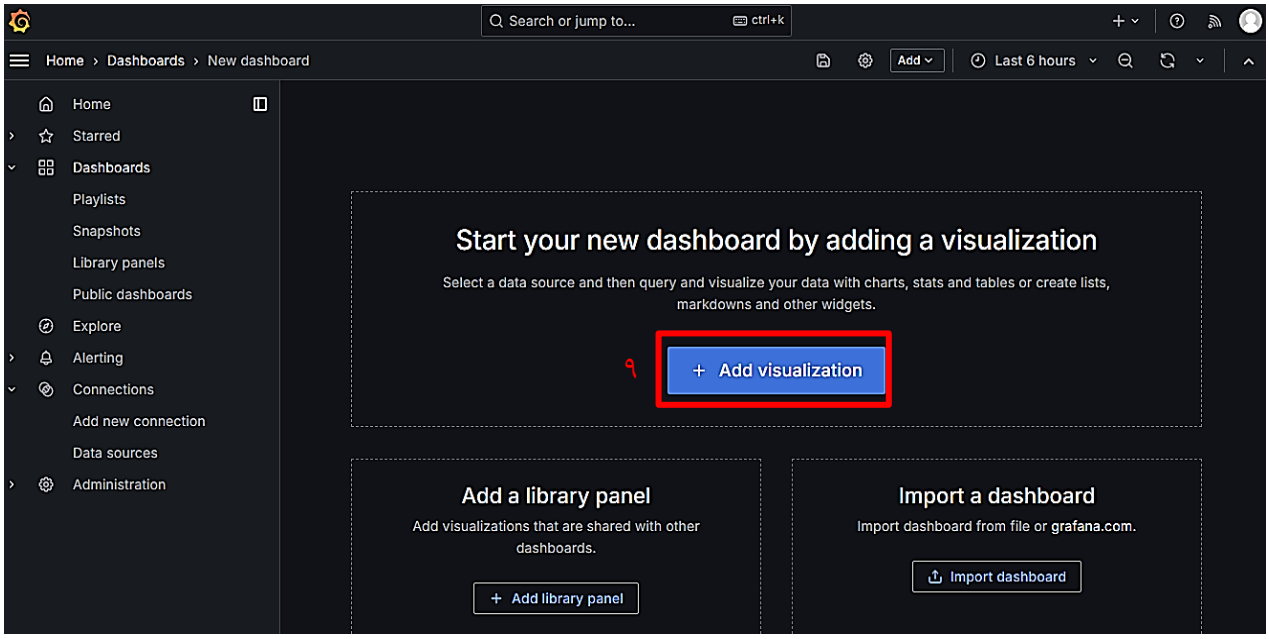
مرحله ۴: ساخت داشبورد

برای ورود به محیط ساخت داشبورد، مطابق شکل ۴-۷ از منوی سمت چپ، روی Dashboards (کادر شماره ۷) کلیک کرده و گزینه Create dashboard (کادر شماره ۸) انتخاب می‌شود، همچنین می‌توان از مسیر New → New Dashboard نیز به همین بخش وارد شد (کادرهای شماره ۸-۱ و ۸-۲):



شکل ٤-۷ – ساخت داشبورد جدید در Grafana

در ادامه مطابق شکل ۴-۸ با انتخاب گزینه Add a visualization پنجره ایجاد پنل جدید باز می‌شود. در این محیط، ابزارهای لازم برای تعریف و نمایش متریک‌ها در اختیار کاربر قرار می‌گیرد و می‌توان پنل‌های مختلف را به داشبورد اضافه کرد.

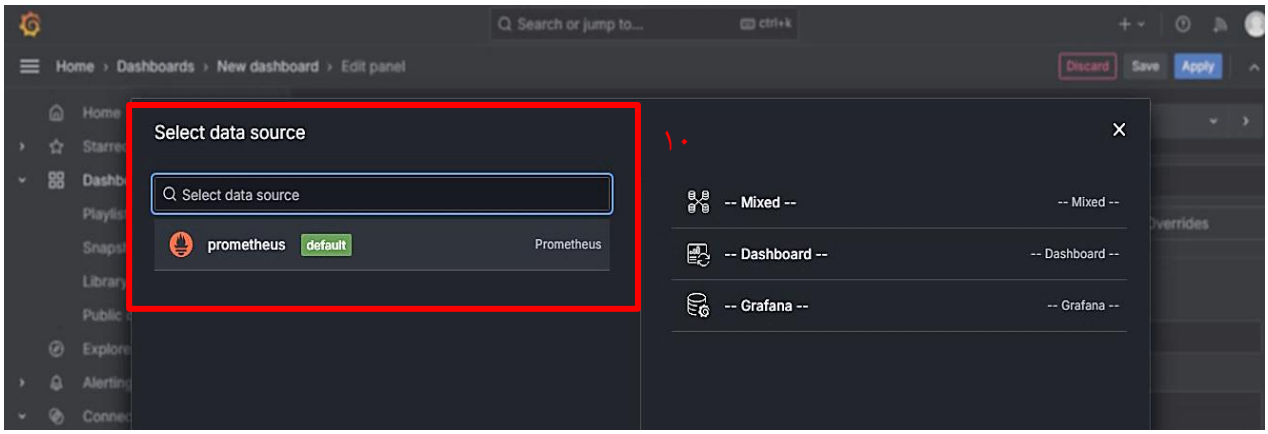


شکل ٤-۸ – ساخت پنل جدید در داشبورد Grafana

مرحله ۵: اضافه کردن منبع داده

در این مرحله، منبع داده‌ای که قرار است متریک‌ها از آن دریافت شوند، مشخص می‌شود.

پس از انتخاب گزینه Add a visualization، محیط ایجاد پنل جدید باز می‌شود. در صورتی که چند منبع داده در Grafana تعریف شده باشد، از فهرست منابع داده، گزینه Prometheus که در مرحله ۳: پیکربندی شده است انتخاب می‌شود. در غیر این صورت، Grafana به‌طور خودکار همان منبع داده را انتخاب می‌کند (شکل ۴-۹).



شکل ٤-۹ – انتخاب Prometheus به عنوان منبع داده

پس از انتخاب منبع داده، محیط ویرایش پنل ظاهر می‌گردد. در این صفحه، در قسمت بالایی، بخش Visualization برای نمایش نتایج و در قسمت پایینی، بخش Query Editor برای ورود و ویرایش عبارت‌های پرس‌وجو قابل مشاهده است.

مرحله ۶: نگارش پرس‌وجو با استفاده از زبان PromQL

در این گام، باید نحوه فراخوانی متریک‌ها از Prometheus مشخص شود. برای این منظور از زبان PromQL[۹۶] استفاده می‌شود که امکان انتخاب، فیلتر، گروه‌بندی، تجمیع و انجام محاسبات روی داده‌های سری زمانی را فراهم می‌کند. در بخش Query Editor، پس از انتخاب منبع داده، دو روش برای ایجاد پرس‌وجو وجود دارد؛ در روش اول می‌توان با استفاده از Query Builder و انتخاب متریک‌ها و عملگرهای موردنیاز، عبارت PromQL را به‌صورت گرافیکی ایجاد کرد (شکل ۴-۱۰). در روش دوم نیز می‌توان همانند شکل ۴-۱۱ عبارت PromQL را به‌صورت مستقیم در محیط ویرایشگر وارد نمود.

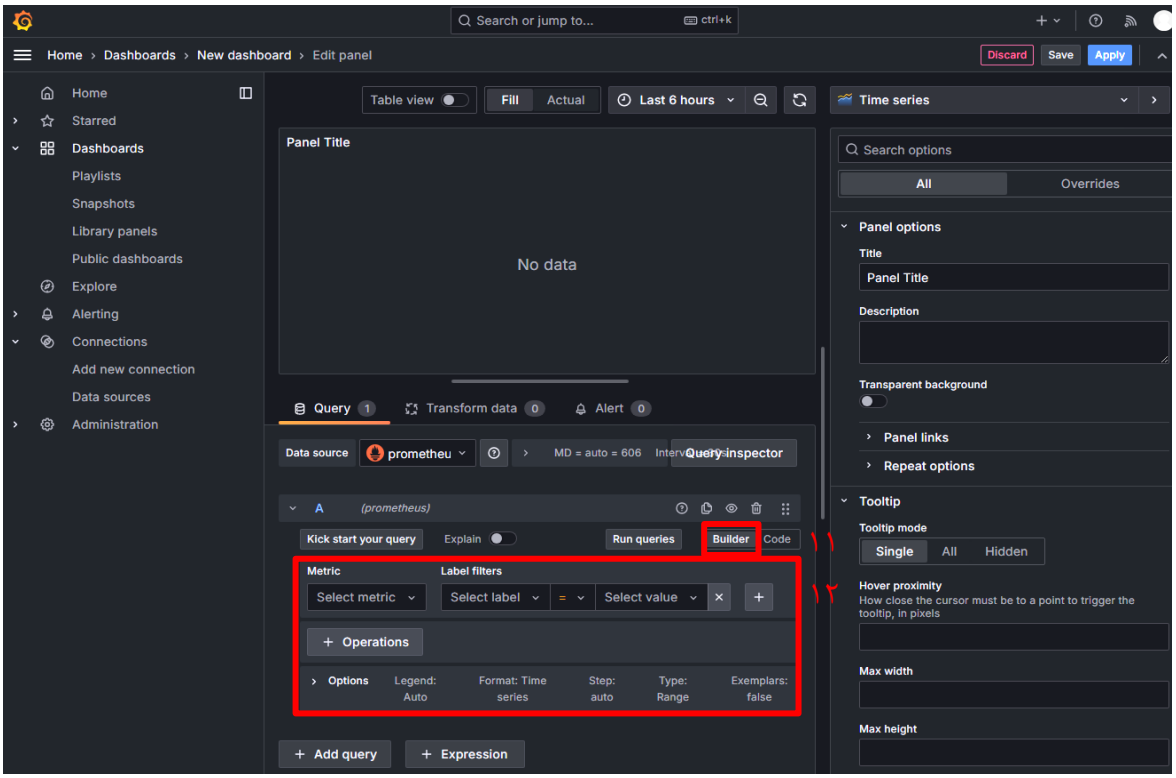
به‌عنوان نمونه، برای نمایش میانگین درصد استفاده از پردازنده، می‌توان از عبارت زیر استفاده کرد؛ این پرس‌وجو، با استفاده از متریک‌های تولید شده توسط Node Exporter، میانگین درصد استفاده از پردازنده را برای هر instance[۹۷] در بازه زمانی پنج دقیقه گذشته محاسبه می‌کند.

```
100- (avg by (instance) (rate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
```

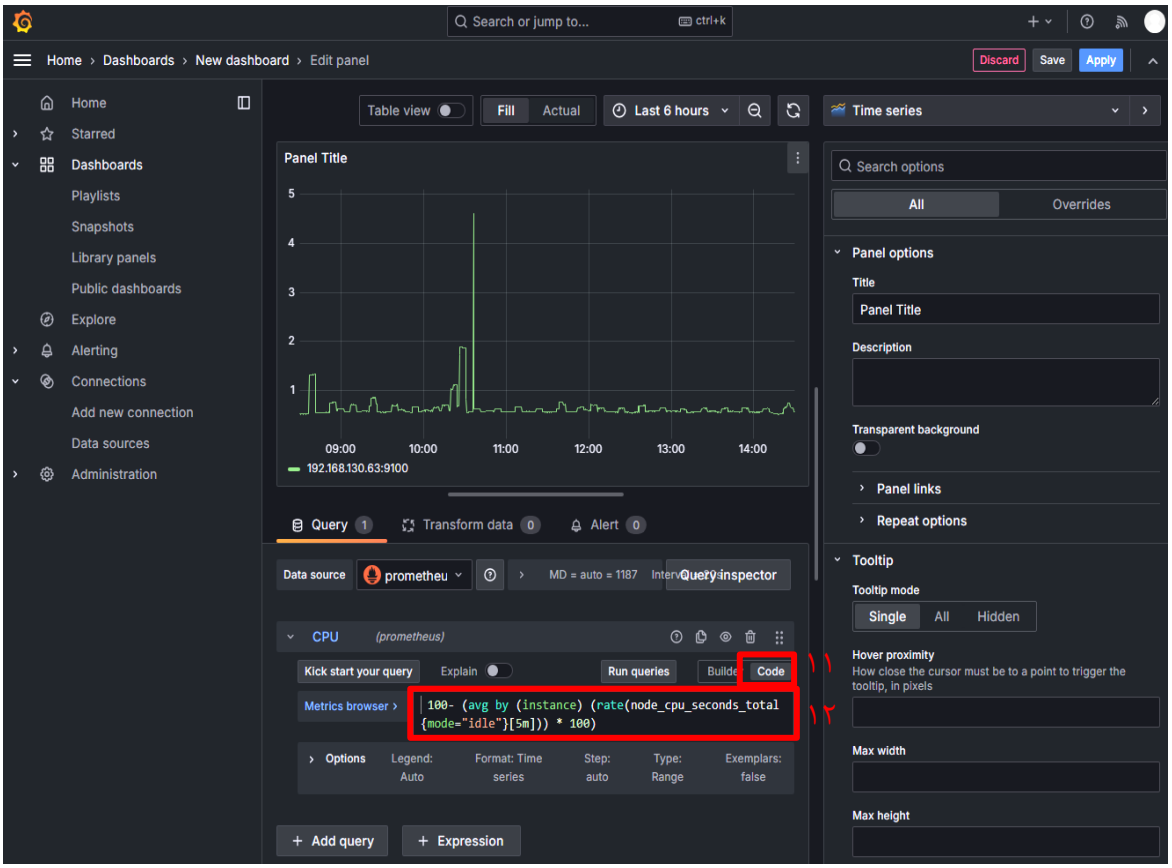
---

[۹۶] Prometheus Query Language

[۹۷] برچسبی در Prometheus که معمولاً آدرس و پورت هر Target را مشخص می‌کند.

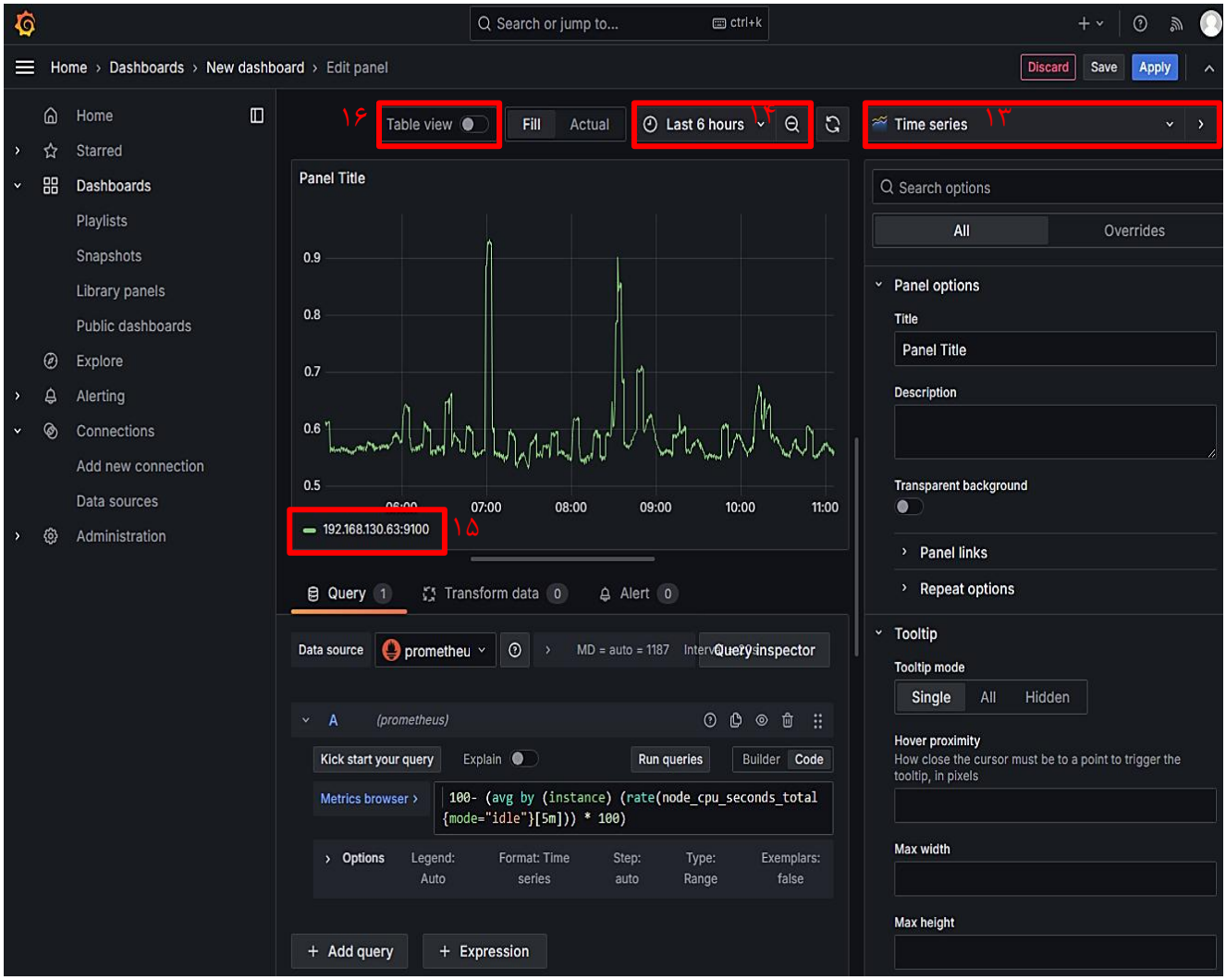


شکل ٤-١٠ – محیط ساخت عبارت PromQL در قسمت Query



شکل ٤-١١ – وارد کرد کد PromQL در قسمت Query

با انتخاب یا نگارش عبارت پرس‌وجوی مورد نظر، با کلیک بر روی گزینه Run queries، داده‌ها از Prometheus واکشی شده و نتیجه آن در قسمت بالایی پنل به‌صورت نمودار نمایش داده می‌شود. در این بخش، امکان تغییر نوع نمایش داده‌ها (مانند Line, Bar, Gauge و سایر نمودارها)، بازه زمانی و سایر تنظیمات نمایشی وجود دارد. به‌عنوان مثال از کادر شماره ۱۳ می‌توان نوع نمودار را تغییر داد، از کادر شماره ۱۴ می‌توان بازه زمانی را تنظیم کرد، از کادر شماره ۱۵ می‌توان رنگ نمودارها را تنظیم نمود و از کادر شماره ۱۶ بین نمایش نموداری (Graph) و جدولی (Table) متریک‌ها جابه‌جا شد. علاوه بر این، تنظیمات دیگری نیز برای سفارشی‌سازی ظاهر و نحوه نمایش پنل در دسترس است (شکل ۴-۱۲).



شکل ٤-۱۲ – گزینه‌های تنظیمات بصری نمودارهای Grafana

در پایان با گزینه save از سمت راست بالای پنجره و تعیین نام دلخواه برای پنل، داشبورد ذخیره می‌شود.

همانند مثال قبل که مربوط به Node Exporter بود، در ادامه نمونه‌ای از متریک‌های Tomcat Exporter نیز ارائه شده است. متریک اول، مقدار حافظه‌ی Heap مصرف‌شده توسط JVM را بر حسب بایت نشان می‌دهد. منظور از Heap بخشی از حافظه است که اشیا و داده‌های مورد استفاده برنامه در زمان اجرا در آن ایجاد و نگهداری می‌شوند؛ برای مثال داده‌هایی که ژئوسرور هنگام پردازش

یک درخواست نقشه ایجاد می‌کند. متریک دوم، میزان حافظه Non-Heap مصرف‌شده توسط JVM را نمایش می‌دهد. این بخش شامل حافظه مورد استفاده برای نگهداری اطلاعات ساختاری ماشین مجازی جاوا، مانند Metaspace، Code Cache و سایر ساختارهای داخلی JVM است. به‌طور خلاصه، Heap عمدتاً برای داده‌ها و اشیای برنامه در زمان اجرا استفاده می‌شود، در حالی که Non-Heap برای نگهداری اطلاعات مربوط به اجرای خود JVM و ساختارهای داخلی آن به کار می‌رود.

```
jvm_memory_bytes_used{area="heap"}
jvm_memory_bytes_used{area="nonheap"}
```

٤-۳- پایش سرویس‌های مکانی

در ادامه‌ی فرآیند پایش سیستم‌ها و سرورها، بخش بعدی به پایش سرویس‌های مکانی اختصاص دارد. در این مستند، سرویس‌های مکانی به سرویس‌هایی اطلاق می‌شود که داده‌ها، لایه‌ها و نقشه‌های مکانی را از طریق بسترهایی مانند ژئوسرور و بر پایه‌ی استانداردهای OGC نظیر WMS، WFS و WCS در اختیار کاربران و سامانه‌های دیگر قرار می‌دهند. ازاین‌رو، بررسی مستمر وضعیت دسترس‌پذیری، صحت پاسخ‌دهی، زمان پاسخ و پایداری این سرویس‌ها برای تداوم عملکرد صحیح سامانه‌های اطلاعات مکانی ضروری است.

یکی از ابزارهای تخصصی برای پایش سرویس‌های مکانی، GeoHealthCheck است. این ابزار به‌صورت مستقل و از طریق ارسال دوره‌ای درخواست‌های استاندارد OGC مانند GetCapabilities، GetMap و GetFeature به سرویس‌هایی نظیر WMS، WFS و WCS عمل می‌کند. GeoHealthCheck پس از ارسال هر درخواست، دسترس‌پذیری سرویس، صحت و اعتبار پاسخ دریافتی، وجود یا نبود خطا و زمان پاسخ‌دهی را بررسی می‌کند. به این ترتیب، پایش سرویس مکانی فقط به روشن بودن سرور یا فعال بودن فرایند نرم‌افزاری محدود نمی‌شود، بلکه عملکرد واقعی سرویس از دید کاربر یا سامانه‌ی مصرف‌کننده نیز ارزیابی می‌گردد.

در ادامه، به راه‌اندازی و پایش GeoHealthCheck پرداخته خواهد شد تا نشان داده شود چگونه می‌توان با استفاده از درخواست‌های استاندارد OGC، سلامت عملیاتی سرویس‌های مکانی را ارزیابی کرد و در کنار داده‌های زیرساختی Prometheus، دید کامل‌تری از وضعیت سرویس و بستر اجرای آن به دست آورد.

۴-۳-۱- نصب GeoHealthCheck

در ادامه، راه‌اندازی و پیکربندی GeoHealthCheck بر روی سرور پایش‌کننده تشریح می‌شود. با توجه به اینکه در بخش قبل نقش این ابزار در پایش سرویس‌های مکانی توضیح داده شد، تمرکز این قسمت بر آماده‌سازی محیط اجرا، نصب وابستگی‌های لازم و فراهم‌سازی دسترسی به داشبورد GeoHealthCheck است. پس از انجام این مراحل، امکان تعریف سرویس‌های مکانی مورد نظر و آغاز پایش آن‌ها فراهم خواهد شد.

مرحله ۱: بررسی python در سرور پایش‌کننده

GeoHealthCheck و کتابخانه‌ها و وابستگی‌های موردنیاز آن بر پایه Python3 اجرا می‌شوند. لذا باید مطمئن شد که Python3 نصب است و نسخه نصب شده بررسی شود:

```
python3 --version                    # بررسی نصب و نسخه python3
```

مرحله ۲: افزودن کتابخانه‌ها و بسته‌های پیش‌نیاز

برای راه‌اندازی GeoHealthCheck بر روی یک ماشین مجازی در لینوکس، پیش‌نیازهای زیر ضروری است:

- pip (مدیریت بسته‌های پایتون)
- git (برای کلون[۹۸] کردن کدهای برنامه)
- python3-venv (برای ایجاد محیط مجازی پایتون)

در صورت دسترسی سرور به اینترنت آزاد، با دستور زیر هر سه بسته‌ی پیش‌نیاز نصب می‌شوند:

```
sudo apt install -y python3 python3-pip python3-venv git    # نصب آنلاین هر سه بسته
```

ولی از آنجا که اغلب سرورها دسترسی به اینترنت آزاد ندارند، بایستی تمامی موارد به‌صورت آفلاین دانلود شده و بر روی سرور منتقل گردند. لذا ابتدای کار یک پوشه (مثلا به نام packs) ایجاد (sudo mkdir packs) و فایل‌ها بصورت متمرکز در این پوشه بارگذاری می‌شوند.

الف) pip: به این منظور فایلی به‌نام get-pip.py را از اینترنت دانلود نموده و یا فایلی با محتویات برابر با لینک زیر با پسوند py. ساخته و به پوشه packs منتقل می‌شود و سپس می‌توان آن را با دستور زیر به سرور اضافه نمود:

https://bootstrap.pypa.io/get-pip.py

```
cd packs                             # ورود به پوشه
python3 get-pip.py                   # نصب pip
```

ب) git: از لینک زیر فایل نصب را دانلود و به پوشه packs منتقل نموده که با دستورات زیر نصب می‌شود:

http://security.ubuntu.com/ubuntu/pool/main/g/git/git_2.34.1-1ubuntu1.15_amd64.deb

```
sudo dpkg -i *.deb                   # نصب فایل‌ها
sudo apt-get install -f              # رفع و نصب وابستگی‌های موردنیاز
```

ج) python3-venv: این ابزار خود پیش‌نیازهایی دارد که بایستی همراه آن دانلود شده و نصب شود:

http://launchpadlibrarian.net/580951501/python3-setuptools-whl_59.6.0-1.2_all.deb
http://launchpadlibrarian.net/697874337/python3-pip-whl_22.0.2+dfsg-1ubuntu0.4_all.deb
http://launchpadlibrarian.net/671653763/python3.10-venv_3.10.12-1~22.04.2_amd64.deb

[۹۸] Clone: دریافت یک نسخه کامل از مخزن کد بر روی سیستم محلی

```
sudo dpkg -i *.deb                    # نصب فایل‌ها
sudo apt-get install -f               # نصب با وابستگی‌ها
```

مرحله ۳: ایجاد محیط مجازی برای نصب و اجرای GeoHealthCheck

برای نصب و اجرای GeoHealthCheck توصیه می‌شود از یک محیط مجازی پایتون استفاده کرد. این کار به ما اجازه می‌دهد تا کتابخانه‌ها و وابستگی‌های مورد نیاز این نرم‌افزار به صورت ایزوله و مستقل از سایر برنامه‌های نصب‌شده در سیستم مدیریت شوند. بدین ترتیب از تداخل نسخه‌های مختلف کتابخانه‌ها یا ایجاد ناسازگاری در پروژه‌های دیگر جلوگیری خواهد شد.

```
python3 -m venv ghc                   # ساخت محیط مجازی
cd ghc                                # ورود به پوشه
source bin/activate                   # فعال‌سازی ماشین مجازی
```

مرحله ۴: دانلود و نصب GeoHealthCheck

برای دانلود این ابزار به لینک زیر مراجعه و از منوی Code گزینه Download ZIP را انتخاب و فایل دانلود می‌شود.

https://github.com/geopython/GeoHealthCheck

فایل دانلود شده درون پوشه ghc بارگذاری و منتقل می‌شود. برای نصب:

```
unzip GeoHealthCheck-master.zip       # خارج کردن از حالت فشرده
mv GeoHealthCheck-master GeoHealthCheck   # تغییرنام پوشه به‌عبارتی ساده‌تر
cd GeoHealthCheck                     # ورود به پوشه
```

مرحله ۵: دانلود و نصب پیش‌نیاز: Paver

یکی از پیش‌نیازهای ضروری نصب GeoHealthCheck ابزار paver است. Paver یک ابزار اتوماسیون برای پروژه‌های پایتون است که وظیفه اجرای و مدیریت فرایندهای نصب، پیکربندی و ساخت پروژه را بر عهده دارد و از نظر عملکرد مشابه ابزارهای make در لینوکس یا ant در Java است.

با فرض فعال بودن محیط مجازی ایجادشده در مرحله قبل، ابزار Paver و وابستگی‌های موردنیاز آن نصب می‌شود. برای نصب آنلاین آن تنها یک دستور pip install Paver وارد می‌شود. ولی در سرور آفلاین، ابزار و وابستگی‌های موردنیاز آن دانلود و به پوشه ghc منتقل و سپس نصب می‌شوند:

https://files.pythonhosted.org/packages/98/1e/37ba8a75bd77ea56a75ef5ae66fe657b79205bbc36556c9456cd641782a4/Paver-1.3.4-py2.py3-none-any.whl
https://files.pythonhosted.org/packages/d9/5a/e7c31adbe875f2abbb91bd84cf2dc52d792b5a01506781dbcf25c91daf11/six-1.16.0-py2.py3-none-any.whl

```
pip install --no-index --find-links=/home/user/ghc six Paver   # نصب آفلاین با whl. دانلودشده
```

مرحله ۶: دانلود و نصب پیش‌نیاز: requirements.txt

این فایل یک لیست متنی از تمامی کتابخانه‌ها و وابستگی‌های نرم‌افزاری مورد نیاز پروژه به همراه نسخه‌های مشخص آن‌ها را در بر دارد. هدف اصلی استفاده از این فایل، تضمین یکپارچگی و سازگاری محیط‌های اجرایی در سیستم‌های مختلف است. به کمک این فایل، مدیریت وابستگی‌ها به شکل ساده‌تری انجام می‌گیرد و از بروز ناسازگاری‌های احتمالی میان نسخه‌های مختلف کتابخانه‌ها جلوگیری خواهد شد.

مطابق توضیحات فوق، فایل requirements.txt فهرست کتابخانه‌ها و وابستگی‌های موردنیاز پروژه را در بر دارد و برخی از این وابستگی‌ها نیز خود به کتابخانه‌های دیگری وابسته هستند. با دستور زیر می‌توان محتوای این فایل و فهرست کتابخانه‌هایی را که pip باید نصب کند مشاهده نمود که همانطور که پیداست تعداد آن‌ها بسیار بالا می‌باشد.

```
cat requirements.txt                    # نمایش محتوی فایل (کتابخانه‌ها و پیش‌نیازها)
```

زمانی که سرور دسترسی به اینترنت دارد تنها با دستور pip install -r requirements.txt فرآیند نصب این بسته و تمامی کتابخانه‌هایش طی می‌شود. ولی در سرور آفلاین، بایستی تمامی پکیج‌های پیش‌نیاز در سیستم دیگری دانلود شده و جهت نصب به روی سرور منتقل گردند. به این منظور در سیستمی که مجهز به پایتون و دسترسی اینترنت می‌باشد، پوشه دانلود شده از مرحله ۴: را کپی کرده و سپس محیط cmd باز و دستورات زیر تایپ می‌شوند:

```
cd …\GeoHealthCheck                     # ورود به پوشه GeoHealthCheck
mkdir packages                          # ساخت پوشه خالی packages
pip download -r requirements.txt -d packages   # دانلود کتابخانه‌ها و پیش‌نیازها درون پوشه ایجاد شده
```

پس از این کار، فولدر packages فشرده شده و بر روی سرور منتقل می‌شود و نهایتاً با دستورات زیر اقدام به نصب requirements.txt می‌شود:

```
cd ghc                                  # ورود به پوشه ghc
unzip packages                          # خارج کردن packages از حالت فشرده
cd packages                             # ورود به پوشه packages
pip install --no-index --find-links=/home/.../packages -r requirements.txt   # نصب آفلاین requirements.txt
```

مرحله ۷: نصب پیش‌نیاز Make

برای اجرای برخی مراحل نصب و پیکربندی GeoHealthCheck، ابزار Make نیز موردنیاز است که از لینک زیر دانلود شده و در پوشه packs موجود در مسیر کاری بارگذاری و سپس اجرا می‌شود:

https://ubuntu.pkgs.org/ubuntu/jammy/main/amd64/make_4.3-4.1build1_amd64.deb

```
cd packs                                # ورود به پوشه packs
sudo dpkg -i make_4.3-4.1build1_amd64.deb   # نصب make
```

مرحله ۸: تنظیمات و راه‌اندازی اولیه GeoHealthCheck

بعد از نصب وابستگی‌ها و آماده‌سازی محیط مجازی، برای راه‌اندازی اولیه GeoHealthCheck از دستور paver setup استفاده می‌شود. ولی در سرور بدون اینترنت، لازم است که فایل‌های بخش رابط کاربری[۹۹] و وابستگی‌های HTML/CSS/JS به صورت محلی در دسترس باشند. این پیش‌نیازها شامل قالب‌های مدیریتی، کتابخانه‌های جاوااسکریپت و پلاگین‌ها هستند که از لینک‌های زیر دانلود می‌شوند:

https://github.com/StartBootstrap/startbootstrap-sb-admin-2/archive/v3.3.7+1.zip
https://github.com/select2/select2/archive/refs/tags/4.0.3.zip
https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.9.4/leaflet.min.js
https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.2/html5shiv.min.js
https://cdnjs.cloudflare.com/ajax/libs/respond.js/1.4.2/respond.min.js

تمامی فایل‌های دانلود شده درون پوشه‌ای (برای مثال با نام assets) قرار گرفته و این پوشه در مسیر ghc/GeoHealthCheck/instance بارگذاری می‌شود.

در قدم بعدی بایستی مسیر استفاده از این فایل‌ها در فایل pavement.py پیکربندی شود. بدین‌منظور بهینه است که فایل pavement.py موجود در مسیر ghc/GeoHealthCheck مطابق با محتویات پیوست ۱: متن فایل pavement.py ویرایش شود؛ یا می‌توان فایل قبلی را حذف نموده و فایل pavement.py جدید را در رایانه شخصی بر اساس پیوست ۱: متن فایل pavement.pyساخته و در همان مسیر بارگذاری کرد. در این ویرایش، فایل pavement.py به‌گونه‌ای پیکربندی می‌شود که فایل‌های رابط کاربری را به‌جای دریافت از اینترنت، از پوشه assets بارگذاری کند.

در این مرحله، با اجرای دستور زیر، تنظیمات اولیه GeoHealthCheck انجام شده، پایگاه داده اولیه ایجاد می‌شود، فایل‌های موردنیاز سامانه آماده می‌شوند و محیط اجرای نرم‌افزار برای استفاده آماده می‌گردد:

```
paver setup                                    # اعمال تنظیمات GeoHealthCheck
```

مرحله ۹: تولید کلید امنیتی

در این مرحله، یک کلید امنیتی برای GeoHealthCheck تولید می‌شود. این کلید که معمولاً با عنوان Secret Key شناخته می‌شود، برای انجام برخی عملیات امنیتی داخلی برنامه، از جمله مدیریت نشست کاربران، امضای نشست‌های کاربران و داده‌های ذخیره‌شده در مرورگر و جلوگیری از دستکاری درخواست‌ها مورد استفاده قرار می‌گیرد. کلید تولیدشده باید کپی شده و در محلی امن نگهداری شود، زیرا در مراحل بعدی برای تکمیل تنظیمات برنامه به آن نیاز خواهد بود.

---

[۹۹] Front-end

```
paver create_secret_key                                    # تولید کلید امنیتی
```

مرحله ۱۰: پیکربندی GeoHealthCheck

جهت تعریف عوامل و تنظیم بر اساس محیط کاری شخصی در سرور، فایل config_site.py موجود در پوشه instance به‌صورت زیر ویرایش می‌شود:

```
nano instance/config_site.py                               # ویرایش فایل پیکربندی
SQLALCHEMY_DATABASE_URI                                    # آدرس دیتابیس که می‌تواند مقدار پیش‌فرض بماند
SECRET_KEY = ****                                          # مقدار کلید از مرحله ۹ وارد می‌شود
GHC_ADMIN_EMAIL                                            # ایمیل مدیر که می‌تواند مقدار پیش‌فرض بماند
GHC_SITE_TITLE                                             # عنوان سایت داشبورد که می‌تواند مقدار پیش‌فرض بماند
GHC_SITE_URL= 'http://IP:8000'                             # آدرس سایت پیش‌فرض
GHC_REQUIRE_WEBAPP_AUTH                                    # اگر مقدار آن True باشد، دسترسی به رابط وب، تنها
                                                           # برای کاربران احراز هویت‌شده ممکن است.

BABEL_DEFAULT_LOCALE = 'fa'                                # نظیمات زبان و ناحیه زمانی
BABEL_DEFAULT_TIMEZONE = 'Asia/Tehran'
LANGUAGES = {
   'en': 'English',
   'fa': 'فارسی' }
```

مرحله ۱۱: راه‌اندازی پایگاه داده GeoHealthCheck و تعریف نام کاربری و رمز عبور

در این مرحله، پایگاه داده داخلی GeoHealthCheck راه‌اندازی می‌شود. این پایگاه داده برای ذخیره اطلاعاتی مانند کاربران سامانه، سرویس‌های مکانی تعریف‌شده برای پایش، تنظیمات بررسی‌ها، نتایج پایش و تاریخچه وضعیت سرویس‌ها مورد استفاده قرار می‌گیرد. با اجرای دستور زیر، جداول اولیه پایگاه داده ایجاد شده و سپس نام کاربری، رمز عبور و ایمیل مدیر سامانه دریافت می‌شود. پس از تکمیل این مرحله، می‌توان با همین نام کاربری و رمز عبور وارد محیط مدیریت GeoHealthCheck شد.

```
python3 GeoHealthCheck/models.py create                    # راه‌اندازی پایگاه داده و مشخصات کاربر
```

مرحله ۱۲: راه‌اندازی دستی GeoHealthCheck

برای اجرای دستی GeoHealthCheck، ابتدا متغیر زمانی ترمینال روی منطقه زمانی تهران تنظیم می‌شود. سپس فایل اصلی برنامه با Python اجرا می‌گردد. در صورت اجرای دستور بدون تعیین آدرس و پورت، برنامه با تنظیمات پیش‌فرض اجرا می‌شود؛ اما در صورت نیاز می‌توان آدرس IP و پورت مورد نظر را نیز هنگام اجرا مشخص کرد.

```
export TZ='Asia/Tehran'                                    # تنظیم متغیر تایم زون ترمینال
python3 GeoHealthCheck/app.py                              # اجرای سرویس GeoHealthCheck
```

```
python3 GeoHealthCheck/app.py IP:Port          # اجرای سرویس GeoHealthCheck در پورت مشخص[۱۰۰]
```

مرحله ۱۳: اتوماتیک‌سازی راه‌اندازی GeoHealthCheck

به‌منظور اینکه در هر بار نیاز به ابزار یا ریبوت شدن سرور، نیاز به استارت کردن دستی نباشد، فایل سرویس با محتوای زیر ساخته شده و پس از ذخیره و بستن آن، فایل systemd بازخوانی و سرویس فعال می‌شود. در این صورت، پس از هر بار راه‌اندازی مجدد سیستم، سرویس به‌صورت خودکار اجرا شده و نیازی به اجرای دستی مرحله قبل نخواهد بود.

```
sudo tee /etc/systemd/system/geohealthcheck.service > /dev/null <<    # ایجاد فایل سرویس
'EOF'
[Unit]
Description=GeoHealth Check Service
After=network.target

[Service]
User=user
Group=user
WorkingDirectory=/home/user/ghc/GeoHealthCheck
Environment="PATH=/home/user/ghc/bin"
Environment="TZ=Asia/Tehran"
ExecStart=/home/user/ghc/bin/python
/home/user/ghc/GeoHealthCheck/GeoHealthCheck/app.py IP:Port

Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
EOF
sudo systemctl daemon-reload                  # بازخوانی فایل systemd
sudo systemctl enable geohealthcheck          # فعال‌سازی سرویس GeoHealthCheck
sudo systemctl start geohealthcheck           # استارت سرویس GeoHealthCheck
sudo systemctl status geohealthcheck -l       # وضعیت سرویس GeoHealthCheck
```

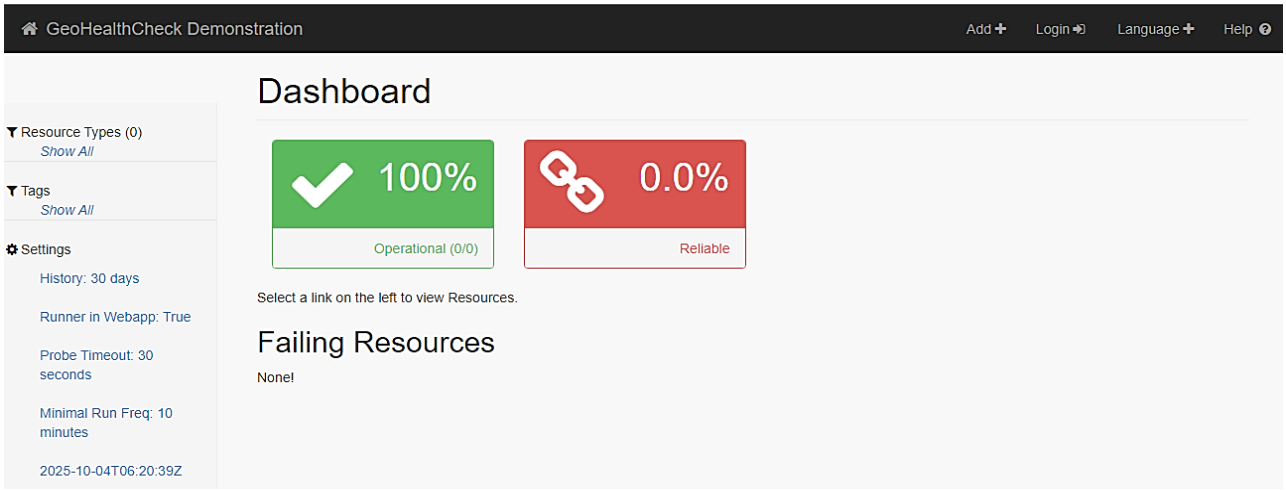
۴-۳-۲- داشبورد GeoHealthCheck

پس از نصب و راه‌اندازی GeoHealthCheck روی سرور پایش‌کننده، رابط وب این سامانه با استفاده از مرورگر در دسترس قرار می‌گیرد. داشبورد GeoHealthCheck به‌منظور نمایش وضعیت سرویس‌های مکانی، نتایج بررسی‌های انجام‌شده، زمان پاسخ، وضعیت

[۱۰۰] نکته: امکان تعیین آدرس IP و پورت در انتهای دستور تنها در صورتی وجود دارد که فایل app.py برای دریافت این پارامترها پیکربندی شده باشد؛ در نسخه استاندارد برنامه، آدرس و پورت باید از طریق فایل پیکربندی یا کد برنامه تعیین شوند.

دسترس‌پذیری و تاریخچه سلامت سرویس‌های مبتنی بر استانداردهای OGC به‌صورت بلادرنگ[۱۰۱] طراحی شده است. این رابط وب در آدرس زیر در دسترس خواهد بود:

http://<IP سرور : Port>



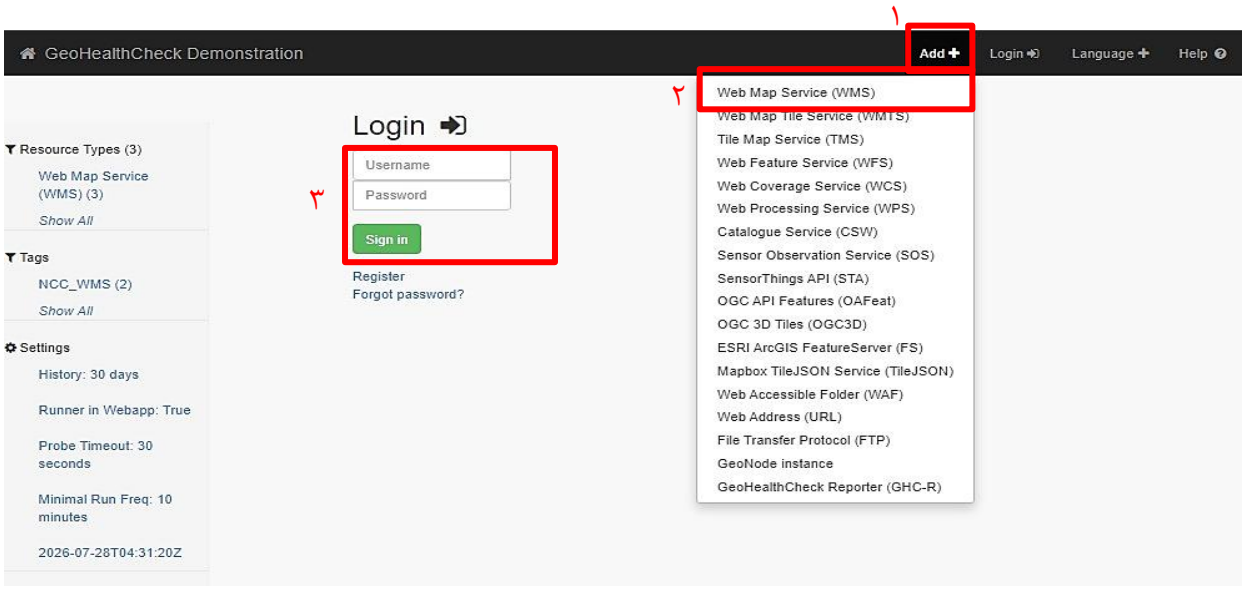
شکل ٤-۱۳ – نمایی از داشبورد GeoHealthCheck

پس از نصب GeoHealthCheck، داشبورد سامانه در وضعیت اولیه قرار دارد و هنوز هیچ سرویس مکانی برای پایش در آن تعریف نشده است؛ بنابراین داده‌ای از سرویس‌های مکانی نمایش داده نمی‌شود. برای اینکه GeoHealthCheck بتواند سرویس‌های مکانی مانند تامکت و ژئوسرور را رصد کند، لازم است چند مرحله تنظیم و پیکربندی انجام شود. (شکل ۴-۱۳)

قبل از شروع هر کاری بایستی مجوز برقراری ارتباط شبکه‌ای بین دو سرور اخذ شود؛ به‌طوری‌که سرور پایش‌کننده به پورت سرویس‌های مکانی در سرور پایش‌شونده دسترسی داشته باشد.

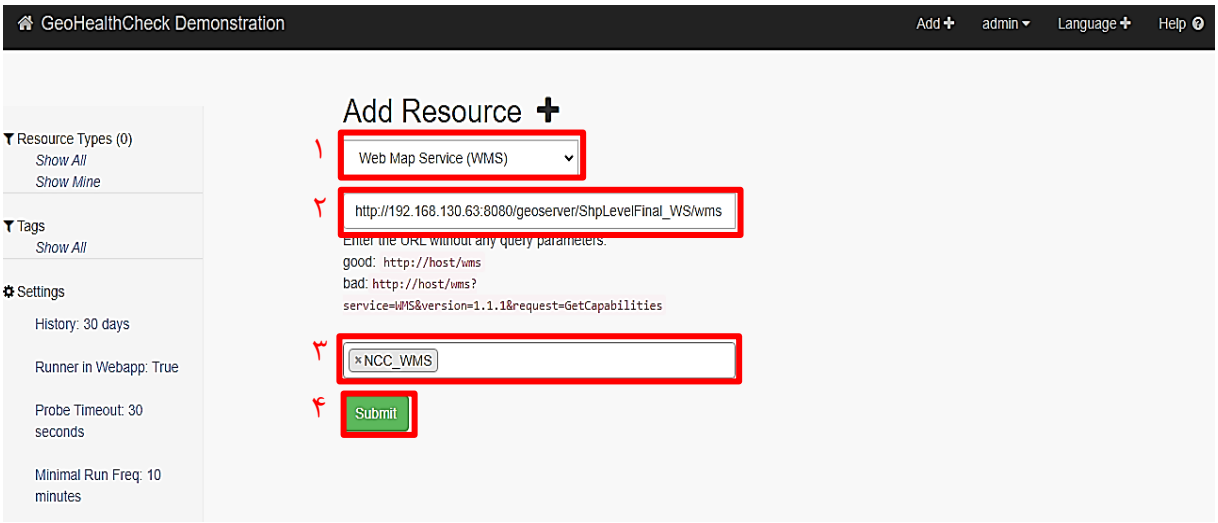
پس از اطمینان از دسترسی سرویس‌ها، می‌توان آن‌ها را از طریق داشبورد وب GeoHealthCheck اضافه کرد. برای این کار، بایستی از منوی Add وارد بخش Add Resource در داشبورد شد که در صورت فعال بودن احراز هویت رابط وب، نام کاربری و رمز عبور مدیر سامانه درخواست می‌شود. این مقادیر برابر با مقادیر وارد شده در مرحله ۱۱: بخش قبل می‌باشند. (شکل ۴-۱۴)

[۱۰۱] real-time

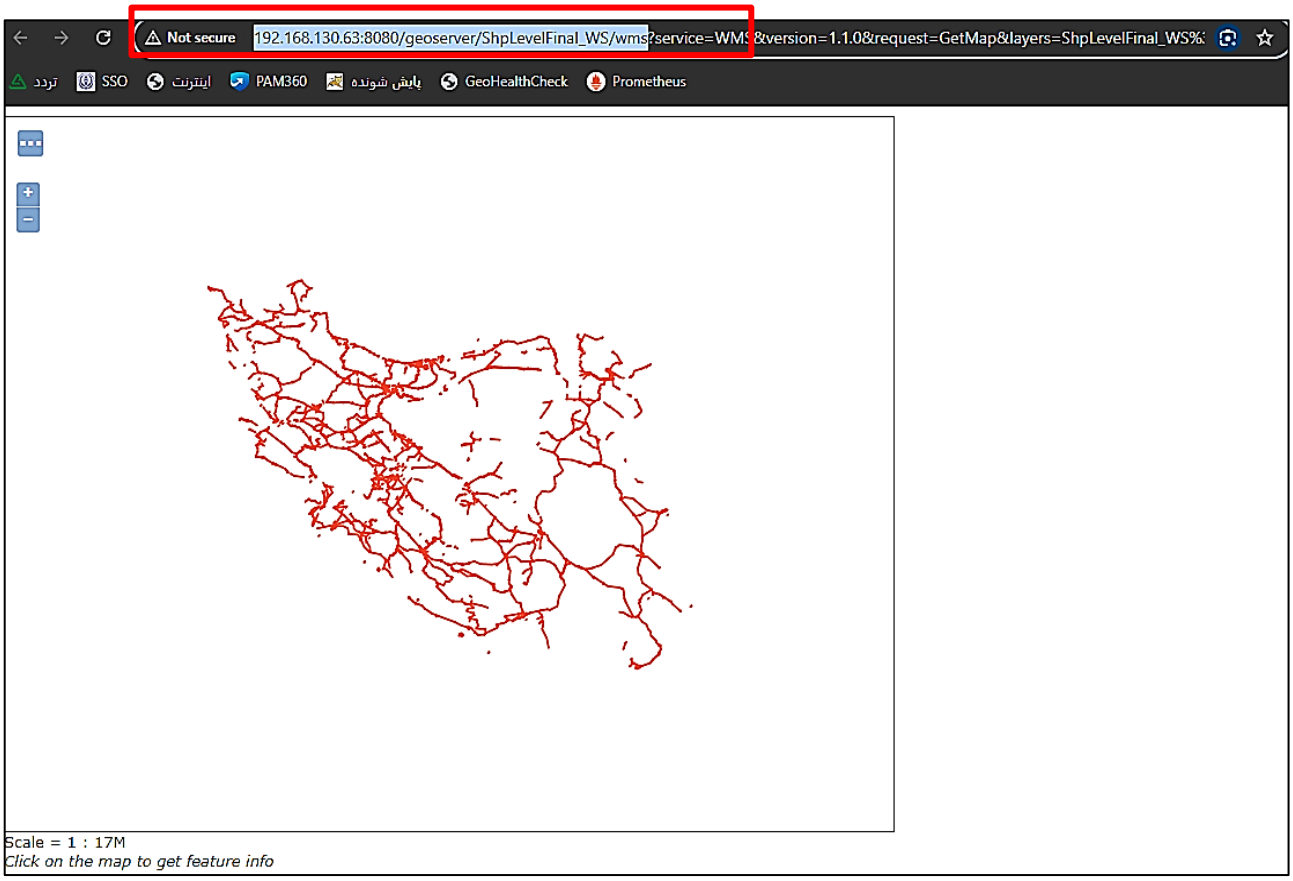


شکل ٤-١٤ – اضافه نمودن WMS جهت رصد در GeoHealthCheck

در پنجره باز شده (شکل ۴-۱۵)، در کادر شماره ۱، نوع سرویس مکانی (همانند WMS) تعیین می‌شود. در کادر شمار ۲ نشانی (URL) سرویس مکانی موردنظر برای پایش وارد می‌شود. باید دقت داشت که نشانی سرویس از آدرس سرویس منتشرشده در GeoServer تا قبل از علامت ? وارد می‌شود (شکل ۴-۱۶). در کادر شماره ۳ برچسب دلخواه برای پایش سرویس مکانی تایپ می‌شود. در پایان با فشردن دکمه Submit، صفحه پیکربندی و تنظیمات منبع پایش نمایش داده می‌شود (شکل ۴-۱۷).



شکل ٤-١٥ – وارد نمودن مشخصات سرویس مکانی هدف پایش در GeoHealthCheck

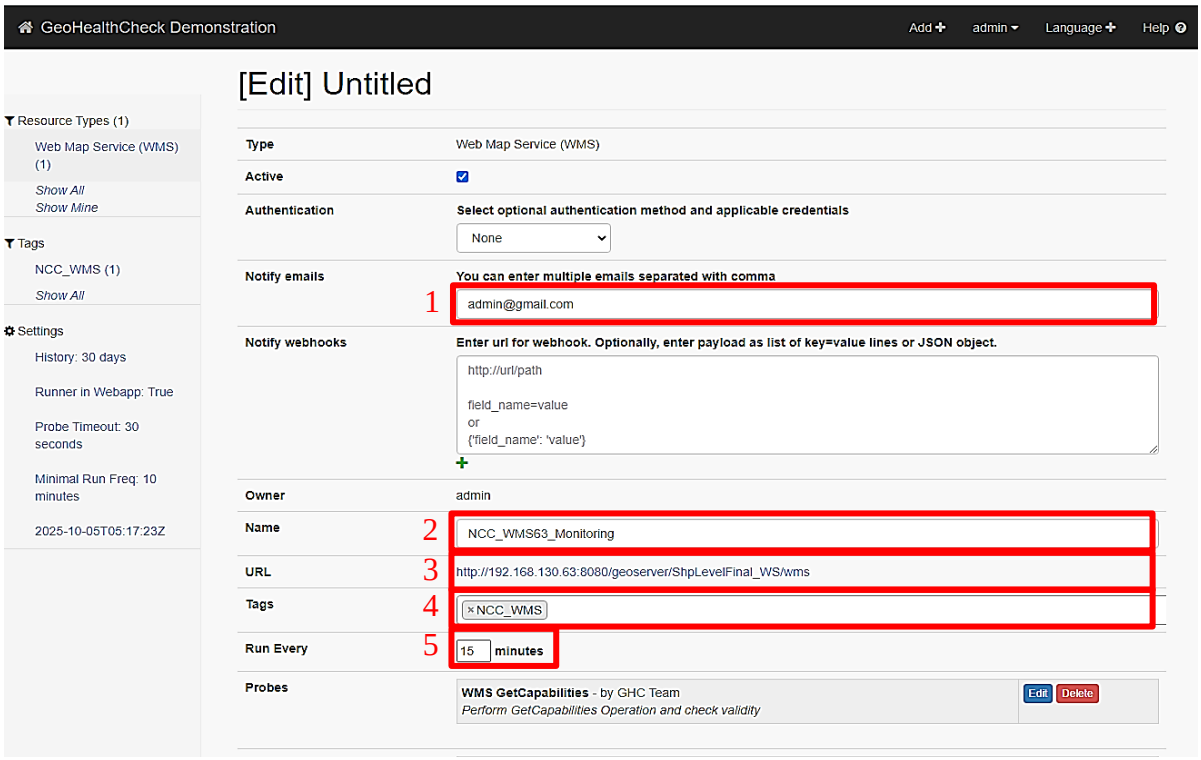


شکل ٤-١٦ – آدرس موردنیاز برای تعریف در پایش برداشت‌شده از ژئوسرور

در پنجره شکل ۴-۱۷، در کادر شماره ۱، نشانی ایمیل کاربر نمایش داده می‌شود. در صورت پیکربندی سرویس ارسال ایمیل و فعال بودن اعلان‌ها، در رخدادهایی مانند از دسترس خارج شدن سرویس، پیام هشدار به این نشانی ارسال خواهد شد. این مقدار در همین بخش قابل ویرایش است.

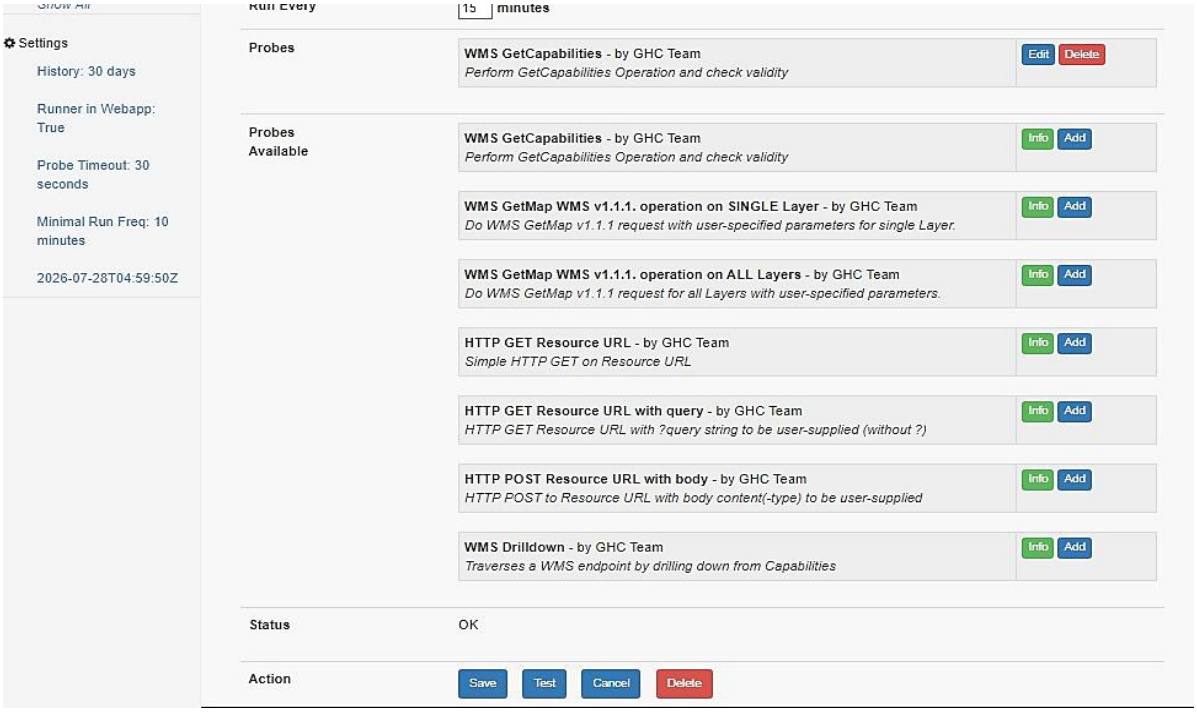
در ادامه در کادر شماره ۲، نامی برای پایش وارد می‌شود. در کادر شماره ۳ که غیرقابل ویرایش است، همان url سرویس مورد پایش (که در گام قبلی وارد شده بود) قابل مشاهده می‌باشد. در کادر بعدی برچسب‌هایی که در پنجره قبل وارد کرده بودیم، نمایش داده می‌شوند و قابل ویرایش هستند.

در کادر شماره ۵ تناوب زمانی پایش سرویس مکانی مشخص می‌شود که CheckGeoHealth بر اساس این مقدار، در فواصل زمانی مشخص، سرویس مکانی را بررسی کرده و نتایج را در پایگاه داده ذخیره می‌کند. برای سرویس‌های حیاتی معمولاً بازه‌های زمانی کوتاه‌تر، مانند یک دقیقه، انتخاب می‌شود؛ اما برای سایر سرویس‌ها می‌توان بازه‌هایی مانند ۱۵ دقیقه یا بیشتر را در نظر گرفت. در مقابل، انتخاب بازه‌های زمانی بسیار کوتاه باعث افزایش تعداد درخواست‌های ارسالی به سرویس‌های مکانی شده و می‌تواند موجب افزایش بار پردازشی سرور، مصرف منابع شبکه و حجم داده‌های ذخیره‌شده در پایگاه داده شود.



شکل ٤-۱۷ – ویرایش تنظیمات پایش سرویس

در ادامه‌ی پنجره، بخش مربوط به Probes ظاهر می‌شود (شکل ۴-۱۸). Probe در GeoHealthCheck به یک آزمون مشخص برای بررسی سلامت سرویس مکانی گفته می‌شود. هر Probe تعیین می‌کند که سامانه چه نوع درخواست یا آزمونی را برای ارزیابی سرویس اجرا کند و چه معیارهایی مانند دسترس‌پذیری، اعتبار پاسخ، زمان پاسخ یا صحت عملکرد سرویس را بررسی نماید.

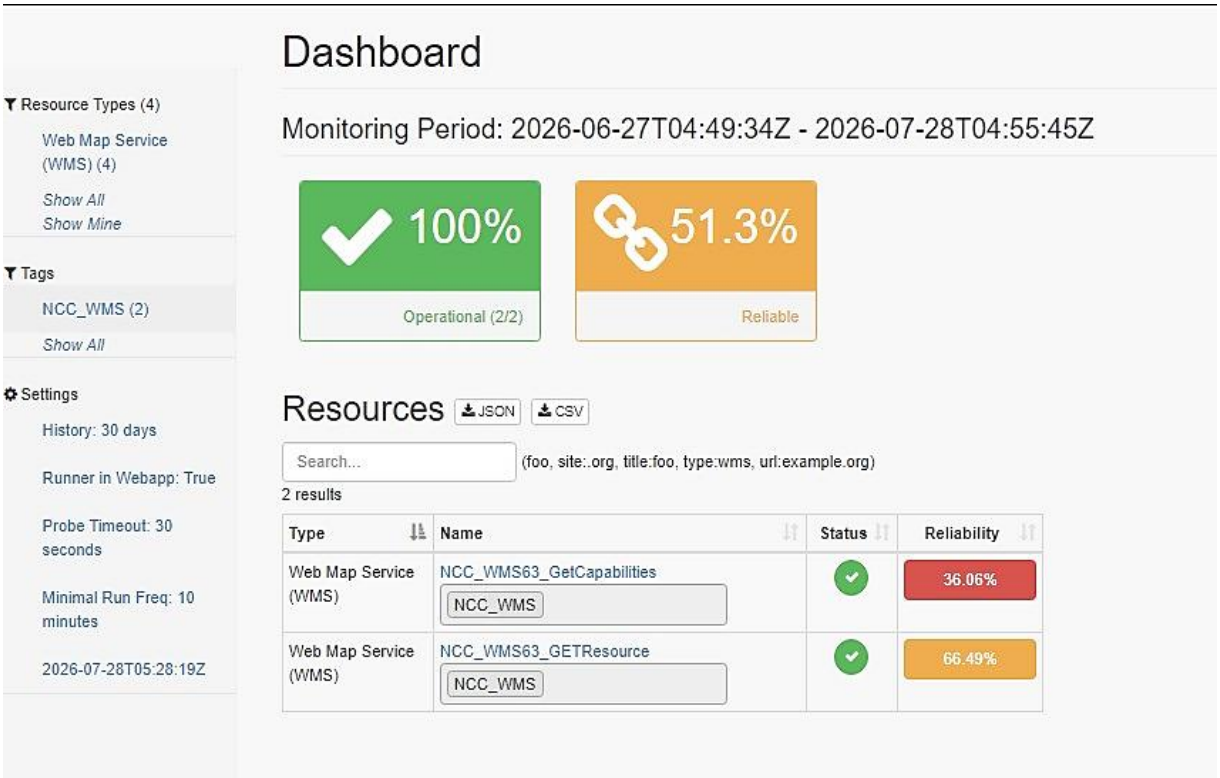


شکل ٤-۱۸– ابزارهای پایش سرویس مکانی در GeoHealthCheck

برای سرویس‌های WMS، معمولاً Probe از نوع WMS GetCapabilities انتخاب می‌شود. این Probe با ارسال درخواست استاندارد GetCapabilities بررسی می‌کند که آیا سرویس در دسترس است، پاسخ معتبر مطابق استاندارد WMS ارائه می‌دهد و اطلاعات توصیفی سرویس، از جمله لایه‌های منتشرشده، نسخه‌های پشتیبانی‌شده و قابلیت‌های آن، قابل دریافت است.

موفقیت این Probe بیانگر صحت پاسخ‌گویی به درخواست GetCapabilities است و به‌تنهایی عملکرد سایر عملیات سرویس، مانند GetMap (که برای تولید و دریافت نقشه به‌کار می‌رود)، را تضمین نمی‌کند. علاوه بر آن، در صورت نیاز می‌توان Probeهای دیگری نیز برای ارزیابی جنبه‌های مختلف عملکرد سرویس تعریف کرد؛ برای مثال، Probe از نوع GetResource با درخواست یک منبع واقعی از سرویس، امکان دسترسی و دریافت صحیح آن منبع را بررسی می‌کند. در صورت نیاز، بسته به نوع سرویس یا هدف پایش، می‌توان Probeهای دیگری نیز انتخاب و به منبع اضافه کرد.

در پایان، تنظیمات ذخیره می‌شوند. پس از انجام این مراحل، GeoHealthCheck سرویس را در بازه‌های زمانی تعیین‌شده به‌صورت خودکار پایش کرده و نتیجه هر بررسی را در داشبورد نمایش می‌دهد. در این داشبورد، رنگ سبز نشان‌دهنده موفقیت‌آمیز بودن آخرین بررسی، رنگ زرد بیانگر وجود هشدار یا نیاز به بررسی بیشتر، و رنگ قرمز نشان‌دهنده ناموفق بودن بررسی یا در دسترس نبودن سرویس است.



شکل ٤-۱۹- نمایی از پایش سرویس‌ها در GeoHealthCheck

به‌عنوان نمونه، آنچه در شکل ۴-۱۹ مشاهده می‌شود، برای یک سرویس WMS دو Probe شامل GetCapabilities و GetResource تعریف شده است. وضعیت هر دو Probe در آخرین بررسی با علامت تیک سبز نمایش داده شده و نشان می‌دهد که

آخرین اجرای آن‌ها موفق بوده است. با این‌حال، مقدار Reliability هر Probe بر اساس نتایج بررسی‌های انجام‌شده در کل بازه زمانی محاسبه می‌شود؛ به همین دلیل، Reliability مربوط به Probe نوع GetCapabilities برابر با ۳۰٫۶۶ درصد و با رنگ قرمز، و Reliability مربوط به Probe نوع GetResource برابر با ۶۶٫۴۹ درصد و با رنگ نارنجی نمایش داده شده است.

شاخص کلی Operational در بالای داشبورد برابر با ۱۰۰ درصد است، زیرا هر دو Probe در آخرین بررسی موفق بوده‌اند. در مقابل، شاخص کلی Reliable برابر با ۵۱٫۳ درصد است و تصویری تجمیعی از میزان موفقیت بررسی‌ها در بازه زمانی انتخاب‌شده ارائه می‌دهد. بنابراین سبز بودن وضعیت فعلی لزوماً به معنای پایداری کامل سرویس در کل دوره نیست و پایین بودن Reliability نشان می‌دهد که بخشی از بررسی‌های قبلی با خطا یا پاسخ نامعتبر مواجه شده‌اند.

فصل پنجم

گزارش‌ها و نمایش نتایج

با تکمیل راه‌اندازی زیرساخت پایش سرویس‌های مکانی سازمان نقشه‌برداری کشور و استقرار ابزارهای Prometheus، Grafana و GeoHealthCheck، مرحله‌ی بعدی، تهیه و تنظیم گزارش‌های تحلیلی و مدیریتی بر پایه‌ی داده‌های جمع‌آوری‌شده است. این گزارش‌ها با هدف ارائه‌ی تصویری دقیق از وضعیت عملکرد سرورها، سرویس‌های مکانی و منابع زیرساختی سازمان تهیه می‌شوند تا مدیران و کارشناسان بتوانند روند عملکرد، تغییرات، نقاط ضعف احتمالی و رخدادهای غیرعادی را شناسایی و مدیریت نمایند.

در این فصل نحوه تبدیل متریک‌ها و نتایج ثبت‌شده به گزارش‌های مدیریتی و قابل‌تحلیل تشریح می‌شود. بخش نخست فصل به گزارش‌های تهیه‌شده از داده‌های Prometheus و داشبوردهای Grafana اختصاص دارد. در این بخش، ابتدا امکان انتخاب و فیلتر سرور پایش‌شونده در داشبورد فراهم شده و سپس گزارش‌های مربوط به دسترس‌پذیری و مقاطع قطعی سرور، فعالیت پردازنده، میزان درگیری حافظه، فضای مصرف‌شده و زمان پاسخ‌گویی دیسک، ترافیک شبکه و دسترس‌پذیری تامکت ایجاد و بررسی می‌شوند. به این ترتیب می‌توان زمان وقوع قطعی‌ها، روند مصرف منابع، افزایش غیرعادی بار سیستم و وضعیت بستر اجرای ژئوسرور را برای هر سرور به‌صورت جداگانه بررسی کرد.

بخش دوم این فصل به گزارش‌های استخراج‌شده از GeoHealthCheck اختصاص دارد که وضعیت در دسترس بودن سرویس‌های مکانی منتشرشده در ژئوسرور و نتایج Probeهای مربوط به درخواست‌های رایج سرویس‌های مکانی بررسی می‌شوند.

۵-۱- گزارش‌های دریافتی از Prometheus و Grafana

این گزارش‌ها بر اساس متریک‌های جمع‌آوری‌شده توسط Prometheus و اطلاعات ارائه‌شده توسط Node Exporter و Tomcat Exporter تهیه می‌شوند و امکان ارزیابی مستمر وضعیت سرورها، منابع سخت‌افزاری و سرویس‌های در حال اجرا را فراهم می‌کنند. در ادامه، نحوه ایجاد پنل‌های مختلف در Grafana و طراحی گزارش‌های مدیریتی بر پایه این متریک‌ها تشریح می‌شود.

مرحله ۱: اضافه کردن Prometheus به Grafana

ورود به Grafana: (http://IP:3000 or 3001).

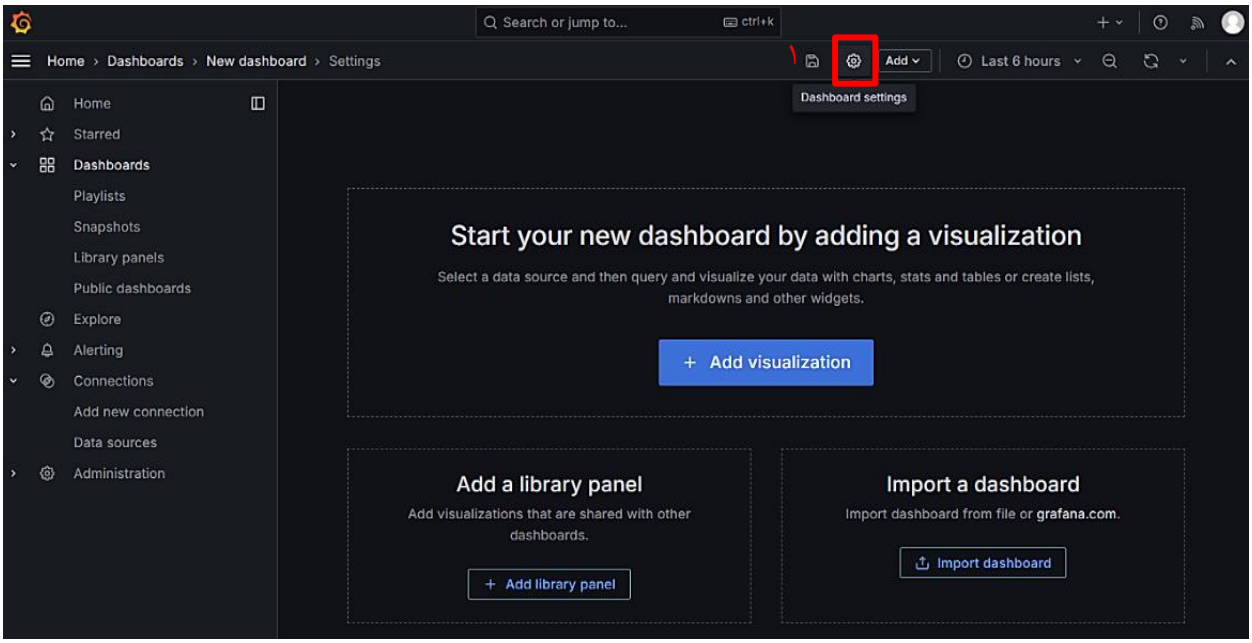
منوی سمت چپ ⟶ Connections → Data Sources → Add new data source

انتخاب Prometheus، سپس در کادر URL آدرس http://IP:9090 تایپ شده و در پایان Save&test.

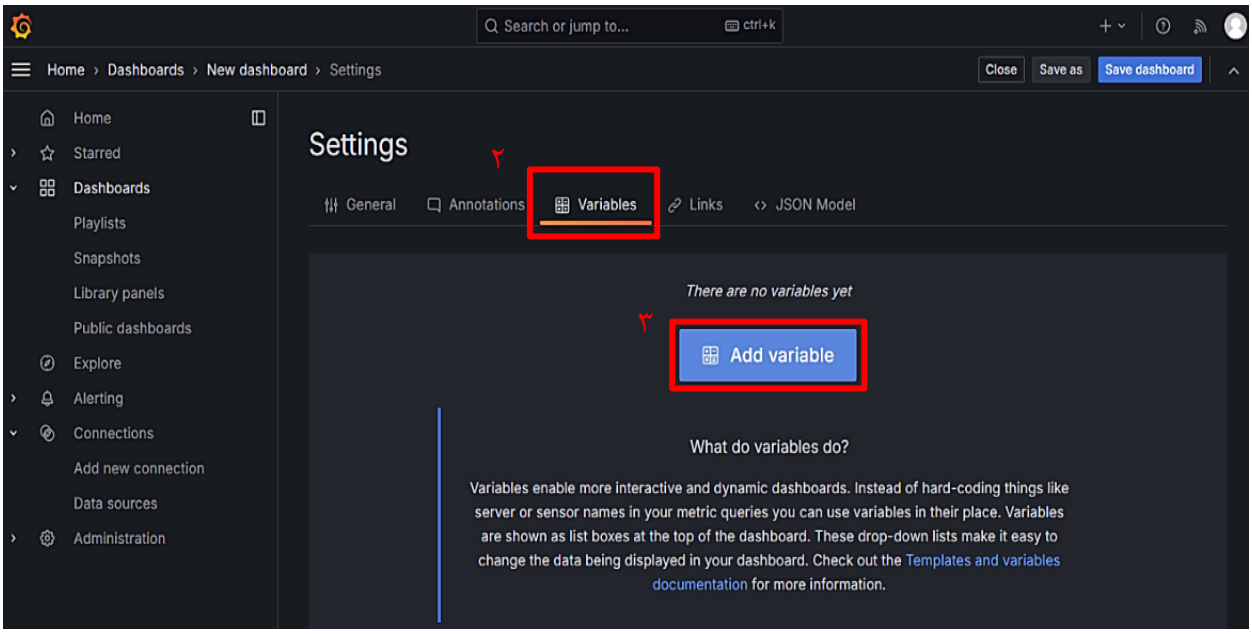
مرحله ۲: ایجاد داشبورد جدید و متغیر instance

برای ایجاد داشبورد همانند شکل ۴-۷ از فصل قبل: Dashboards → New → New Dashboard

در این مرحله به‌منظور استفاده از یک داشبورد برای چندین سرور، یک متغیر instance تعریف می‌شود تا امکان انتخاب سرور موردنظر و تغییر خودکار تمامی پنل‌های داشبورد فراهم گردد. به این منظور بالای صفحه سمت راست، با کلیک بر روی Dashboard settings (نماد چرخ‌دنده) (شکل ۵-۱) و سپس در منوهای ظاهرشده، با انتخاب گزینه‌ی Variables بایستی بر روی Add variable کلیک نمود (شکل ۵-۲).



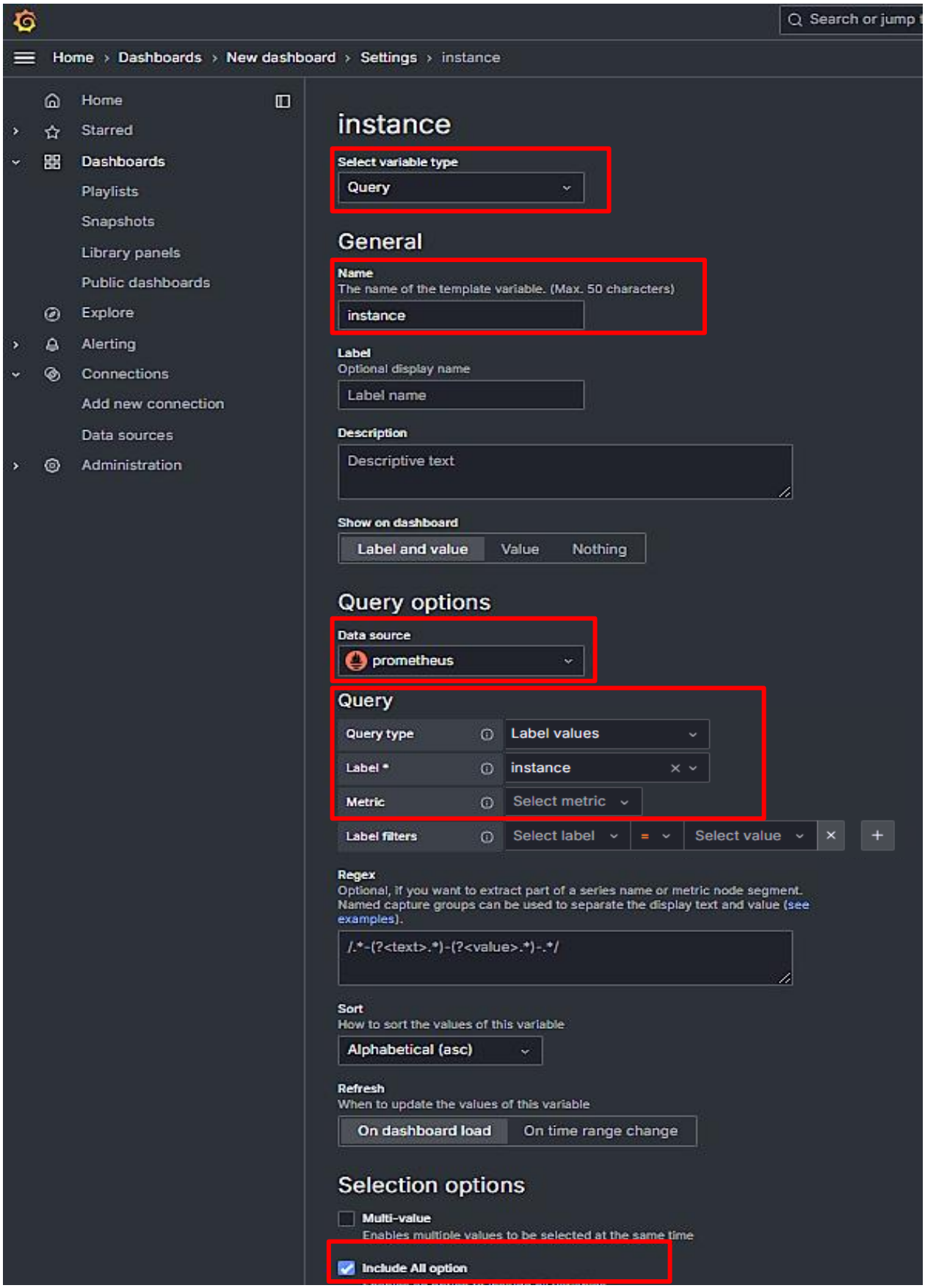
شکل ۵-۱ – تنظیمات داشبورد برای ساخت متغیر



شکل ۵-۲ – ایجاد متغیر برای داشبورد Grafana

در پنجره باز شده همانند شکل ۵-۳ متغیرها به شرح ذیل تنظیم می‌شوند:

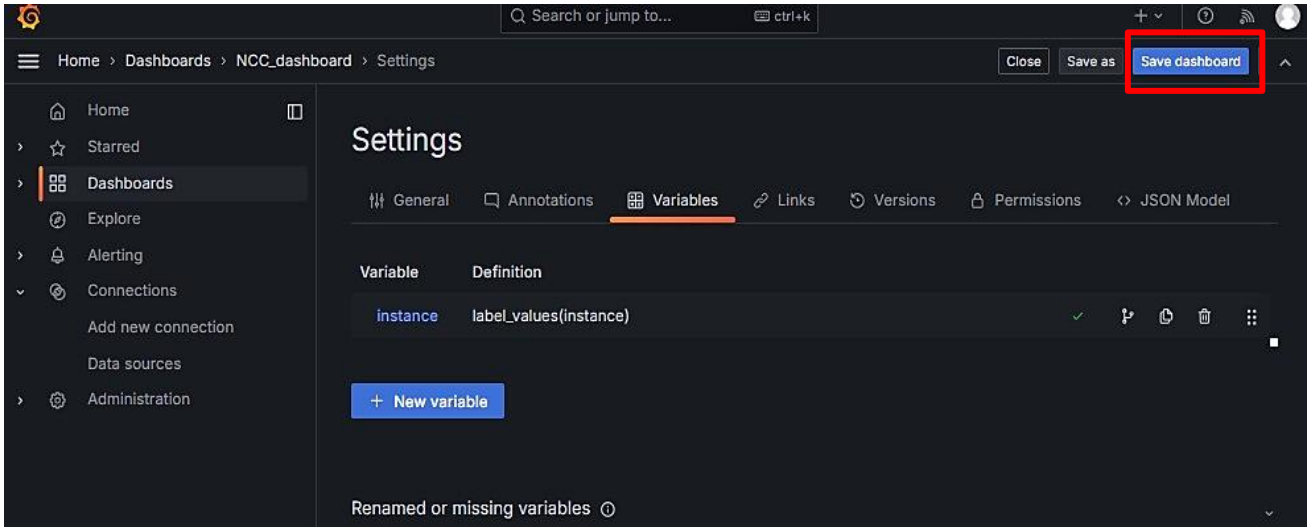
- Query :Select Variable Type
- instance :Name
- Prometheus :Data source
- label values :(Query Type) Query
- instance :(Lable) Query
- (Metric) Query: خالی
- Include All option: فعال



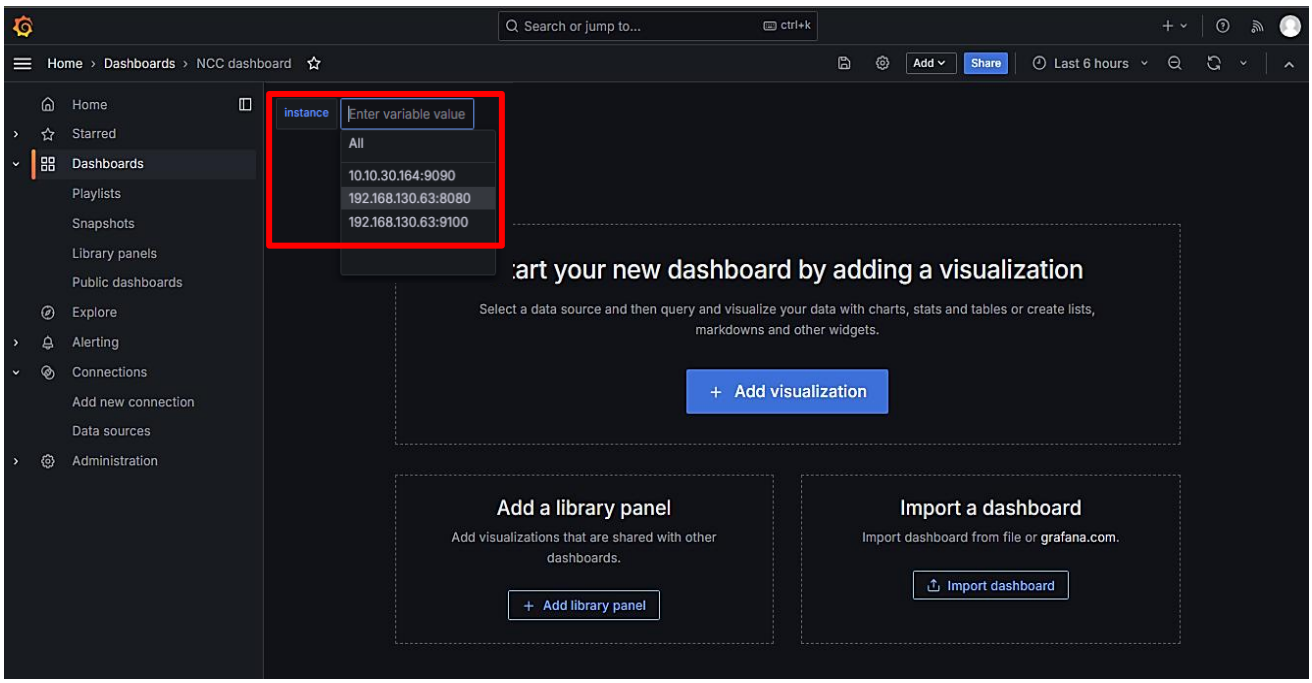
شکل ۵-۳ – تنظیمات تعریف متغیر در داشبورد Grafana

با این تنظیمات، در پایین صفحه، ذیل دکمه‌ی Preview of values فهرست تمامی instanceهای موجود قابل مشاهده می‌باشد. البته، برای نمایش این مقادیر، لازم است سرورها و تارگت‌های مربوطه از قبل در Prometheus تعریف و در حال پایش باشند؛ در پیکربندی انجام‌شده در این مستند، یک سرور شامل سه تارگت برای پایش تعریف شده است.

پس از بررسی مقادیر، با کلیک بر روی دکمه Apply، از قسمت بالای داشبورد، و سپس گزینه Save dashboard انتخاب می‌شود. با وارد کردن نام دلخواه، داشبورد ایجاد و ذخیره می‌شود (شکل ۵-۴). در این مرحله یک منوی کشویی با نام instance به داشبورد اضافه می‌شود که با انتخاب هر instance، تمامی پنل‌های داشبورد به‌صورت خودکار بر اساس همان تارگت فیلتر و به‌روزرسانی می‌شوند (شکل ۵-۵).



شکل ۵-٤- ذخیره داشبورد پس از تعریف متغیر



شکل ۵-۵ – امکان فیلتر پنل‌ها بر اساس IP

در ادامه پنل‌های موردنیاز برای تهیه گزارش‌های اصلی در داشبورد Grafana ایجاد می‌شوند. در گزارش‌هایی که بر پایه متریک‌های Node Exporter هستند، از منوی instance، تارگت مربوط به Node Exporter با پورت ۹۱۰۰ انتخاب می‌شود.

۵-۱-۱- گزارش دسترس‌پذیری[۱۰۲] سرورها

در این گزارش، درصد در دسترس بودن هر سرور در بازه زمانی انتخاب‌شده و گزارشی از مقاطع قطعی سرور ارائه داده می‌شود.

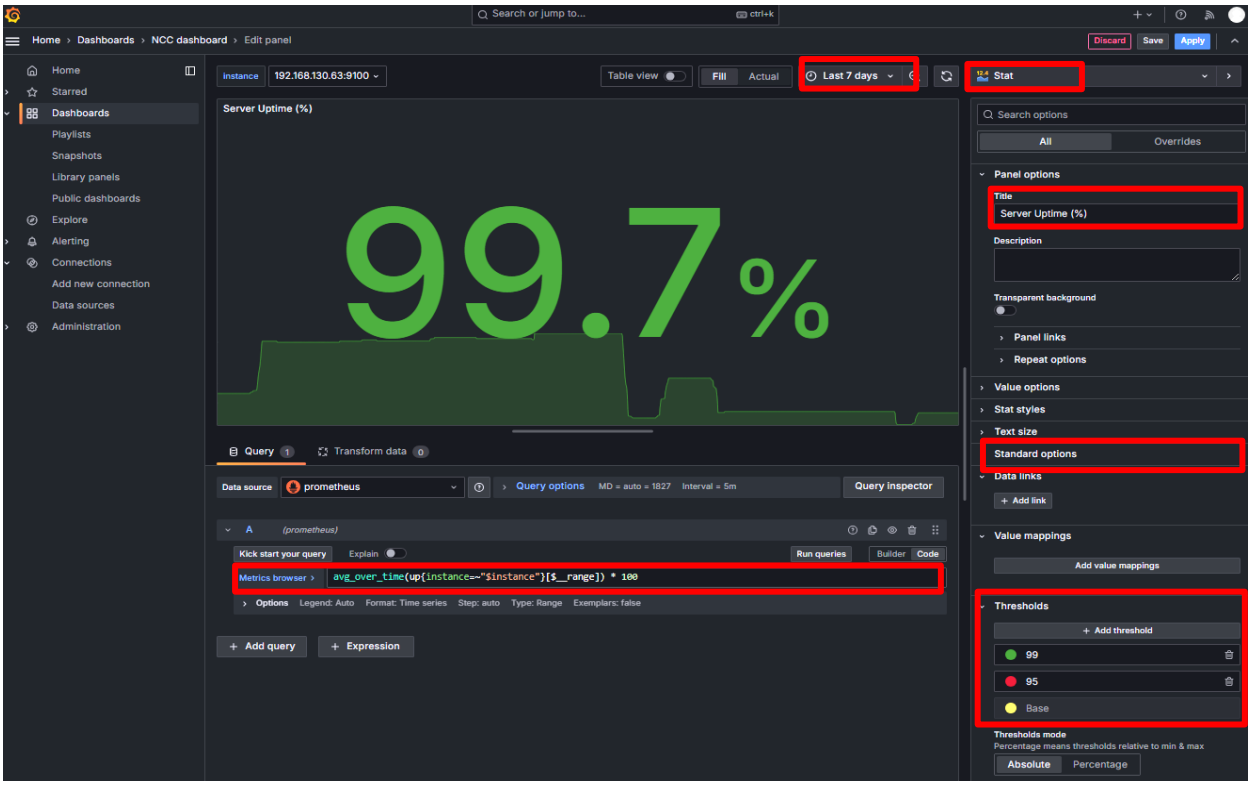
مرحله ۳: ایجاد پنل گزارش درصد دسترس‌پذیری سرور

در صفحه‌ی داشبورد مربوطه، با کلیک بر روی گزینه‌ی Add visualization (شکل ۴-۸ از فصل پیشین) و سپس در قسمت Query options، برای Data source گزینه‌ی Prometheus انتخاب می‌شود (شکل ۴-۹ فصل قبل).

در ادامه، مطابق شکل ۵-۶، در قسمت Query کد زیر وارد می‌گردد:

```
avg_over_time(up{instance=~"$instance"}[$__range]) * 100
```

Visualization را به حالت Stat تغییر داده و بازه پایش را روی ۲۴ ساعت اخیر گذاشته و در Panel title نام « Server Uptime (%) » تعریف می‌شود. در ادامه تنظیمات در قسمت Standard options → Unit گزینه‌ی percent(0-100) را انتخاب کرده و در Thresholds می‌توان مقادیری مانند ۹۹ > را با رنگ سبز و ۹۵ > را با رنگ قرمز نشانه‌گذاری نمود. در پایان با فشردن گزینه Apply، تغییرات ذخیره می‌شود.



شکل ۵-۶ – ایجاد و تنظیم پنل گزارش دسترس‌پذیری سرور

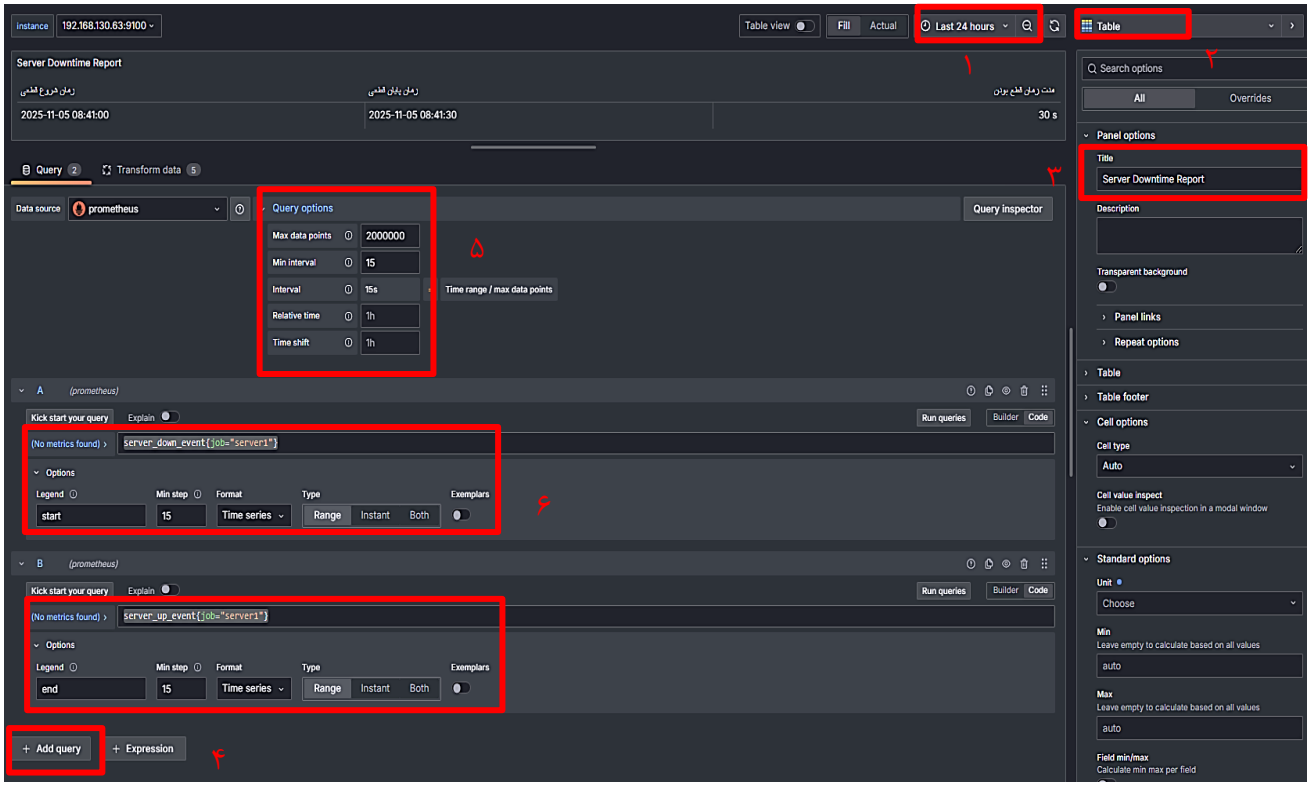
[۱۰۲] Uptime

مرحله ۴: ایجاد پنل فهرست مقاطع قطعی سرور (Downtime Table)

در همان داشبورد پنل جدیدی اضافه کرده (شکل ۵-۷) و در نوار بالای داشبورد، (کادر شماره ۱) محدوده زمانی روی ۲۴ ساعت اخیر تنظیم می‌شود. در سمت راست، (کادر شماره ۲) از Visualization گزینه‌ی Table را انتخاب کرده و در نام پنل (شماره ۳) عبارت دلخواه مانند «Server Downtime Report» وارد می‌شود. در ادامه دو کوئری به شرح ذیل در قسمت کوئری‌ها وارد می‌شود. (برای کوئری دوم دکمه Add query در محل کادر شماره ۴ فشرده می‌شود.)

```
server_down_event{job="server1"}        # Query A
server_up_event{job="server1"}          # Query B
```

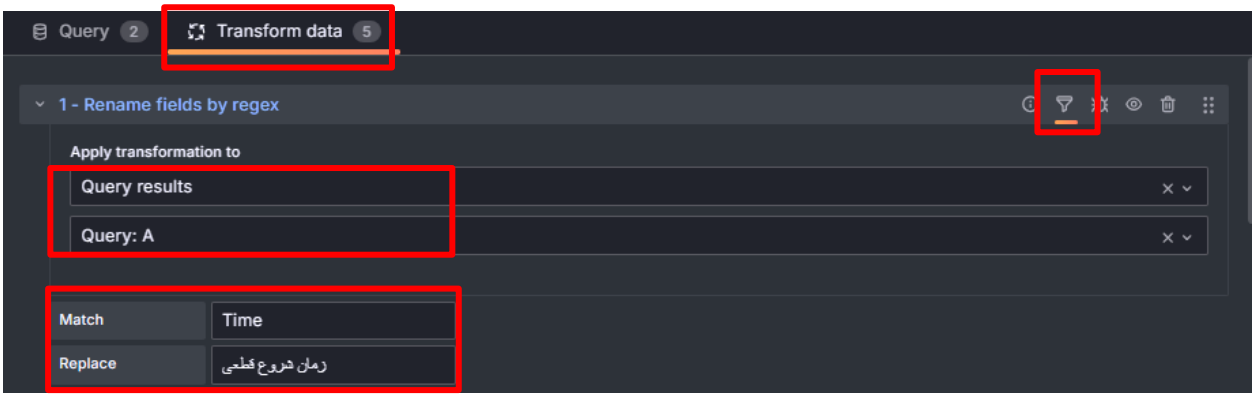
شایان ذکر است قواعد مربوط به این دو کوئری در مرحله ۸: از بخش ۴-۱-۱- تعریف شده‌اند. برای تنظیم وضوح زمانی اجرای کوئری، روی Query options (کادر شماره ۵) کلیک کرده و مقدار Max data points را برابر با ۱۰۰۰۰ و مقدار Min interval برابر با ۱۵ s قرار می‌گیرد. مقدار Max data points حداکثر تعداد نقاطی را مشخص می‌کند که Grafana برای نمایش نتیجه در نظر می‌گیرد و مقدار Min interval نیز اجازه نمی‌دهد فاصله زمانی بین نقاط بازگردانده‌شده از ۱۵ ثانیه کمتر شود. همچنین در قسمت options از هر کوئری (مطابق کادر شماره ۶)، فاصله هر گام (Min step) برابر با ۱۵ ثانیه تعریف می‌شود تا کوئری‌های بازه‌ای با گامی کمتر از فاصله ثبت داده‌ها اجرا نشوند. تأکید می‌گردد این تنظیمات فقط وضوح زمانی دریافت و نمایش نتایج در Grafana را کنترل می‌کنند و جمع‌آوری داده‌ها و ارزیابی قواعد در Prometheus تنظیم و تعریف می‌شوند. در پایان در بخش Legend از کادر شماره ۶، برای کوئری اول عبارت start و برای کوئری دوم عبارت end وارد می‌شود.



شکل ۵-۷ – تعریف کوئری‌ها و تنظیمات مربوط به استخراج قطعی‌های سرور

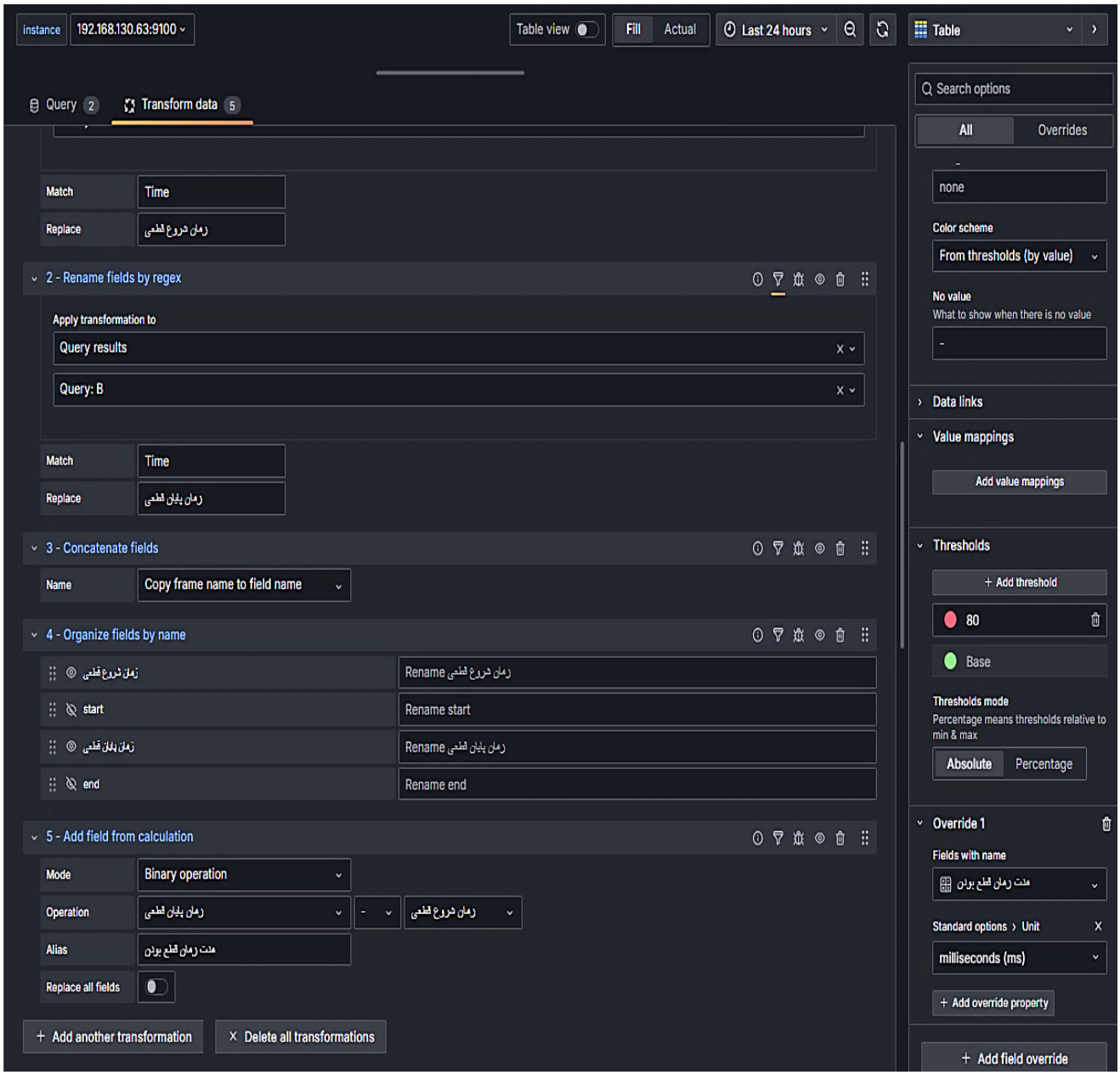
برای استخراج مقاطع قطع سرور، در تب Transform data مراحل زیر به ترتیب اعمال می‌شود:

- تغییر نام زمان کوئری‌ها: به‌این‌منظور از Transform data گزینه Rename fields by regex انتخاب شده و با کلیک بر روی فیلتر، در سطر شماره ۱ Query Results و در سطر شماره ۲، Query:A انتخاب شده و در سطر Match عبارت Time و در سطر Replace عبارت «زمان شروع قطعی» وارد می‌شود. (شکل ۵-۸)



شکل ۵-۸- ترنسفورم تغییر نام فیلدها

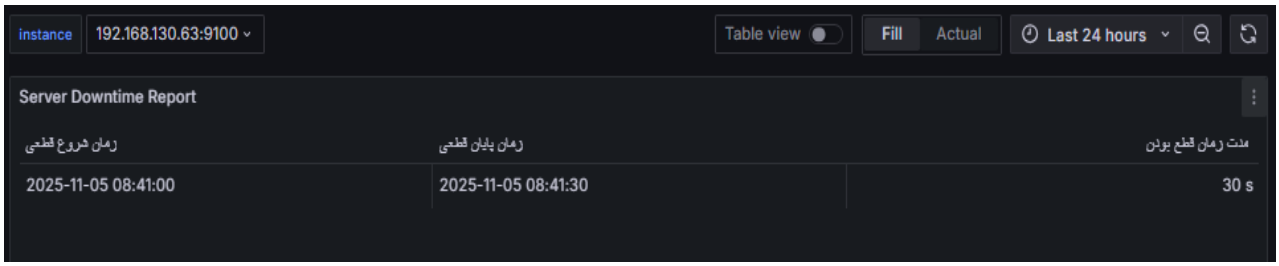
- تغییر نام زمان کوئری‌ها: به‌این‌منظور از پایین پنل Transform data با فشردن دکمه Add another transformation و انتخاب گزینه Rename fields by regex و متعاقبا با کلیک بر روی فیلتر، در سطر شماره ۱ Query Results و در سطر شماره Query:B۲ انتخاب شده و در سطر Match عبارت Time و در سطر Replace عبارت «زمان پایان قطعی» وارد می‌شود.
- روبه‌روی هم قرار دادن زمان‌های شروع و پایان قطعی: به‌این‌منظور با Add another transformation گزینه Concatenate fields انتخاب می‌شود.
- عدم نمایش فیلدهای اضافی: با فشردن دکمه Add another transformation، گزینه Organize fields by name را انتخاب و با کلیک بر روی آیکن 👁 در کنار فیلدهای اضافی، Disable می‌شوند.
- محاسبه مدت زمان قطع بودن: به این منظور با Add another transformation، ترنسفورم Add field from calculation را انتخاب و با قراردادن Mode بر روی Binary operation و سپس در سطر عملیات زمان پایان قطعی منهای (–) زمان شروع قطعی را وارد و در سطر Alias عبارت «مدت زمان قطع بودن» تایپ می‌شود.
- تبدیل مدت زمان محاسبه شده به ثانیه: از تنظیمات سمت راست پنل، با فشردن دکمه Add field override ، Fields with name انتخاب و فیلد با نام «مدت زمان قطع بودن» انتخاب می‌شود. سپس با فشردن دکمه Add override property ، Standards options > Unit انتخاب و نهایتاً در سطر ظاهر شده گزینه time/milisecond (ms) انتخاب می‌شود. (شکل ۵-۹)



شکل ۵-۹ – ترنسفورم‌ها و تنظیمات استخراج مقاطع قطعی سرور پایش‌شونده

برای تست عملکرد پنل، می‌توان در سرور پایش‌شونده کد لینوکسی sudo systemctl reboot را وارد و سرور را ریبوت نمود.

پس از روشن شدن مجدد سرور پایش‌شونده، در پنل Grafana جدولی مانند شکل ۵-۱۰ مشاهده می‌شود.



شکل ۵-۱۰- جدول نمایش مقاطع قطعی سرور پایش‌شونده در پنل Grafana

۵-۱-۲- گزارش فعالیت پردازنده[۱۰۳]

این بخش میزان استفاده از پردازنده را در بازه‌های زمانی مختلف نشان می‌دهد و شامل سه شاخص کلیدی است:

میانگین استفاده (Average Usage) برای سنجش وضعیت عمومی بار سیستم،

بیشینه (Maximum Usage) برای شناسایی لحظات اوج مصرف،

کمینه (Minimum Usage) برای ارزیابی دوره‌های کم‌بار.

این گزارش کمک می‌کند تا ظرفیت سرورها متناسب با حجم کاری تنظیم و در صورت نیاز منابع افزایش یابند.

مرحله ۵: ایجاد پنل گزارش فعالیت پردازنده (CPU)

در همان داشبورد پنل جدیدی اضافه کرده و در قسمت Query کد زیر وارد می‌گردد:

```
100 * (1 - avg by (instance)(rate(node_cpu_seconds_total{mode="idle", instance=~"$instance"}[5m])))
```

از قسمت Visualization، گزینه Stat انتخاب و در بخش Panel title عنوان دلخواه همانند «CPU Usage (%)» را وارد نموده و در قسمت Standard options → Unit، گزینه‌ی percent (0-100) و در قسمت Standard options → Decimals عدد ۲ انتخاب می‌شود.

در قدم بعدی، در بخش پایینی پنل در تب Transform data رفته و با کلیک بر روی Add transformation گزینه‌ی Reduce انتخاب می‌شود. در قسمت Calculation، نیز بایستی گزینه‌های Min و Max و Mean فعال شوند. در بخش Mode of calculation، با انتخاب گزینه‌ی Series to rows هر مقدار در یک ستون جداگانه نمایش داده خواهد شد. برای قالب‌بندی نمایش در بخش Thresholds، رنگ‌های سبز، زرد و قرمز برای محدوده‌هایی همانند مرزهای زیر تنظیم می‌شوند:

تا ۵۰٪: سبز

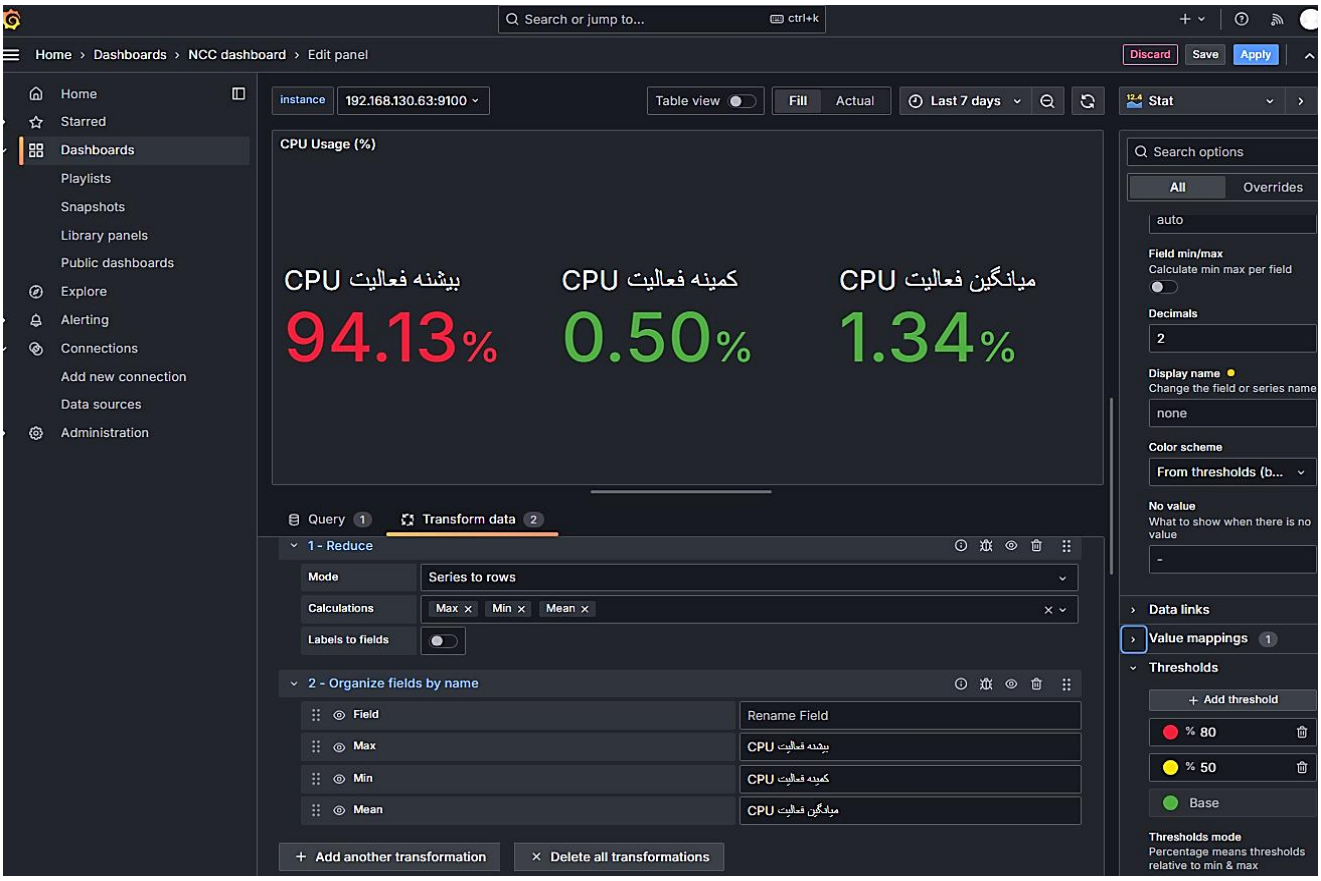
۵۰ تا ۸۰٪: زرد

بالاتر از ۸۰٪: قرمز

برای فارسی‌سازی نام ستون‌ها، با کلیک بر روی Add transformation و انتخاب گزینه‌ی Organize fields by name از فهرست بازشده، در مقابل هر فیلد نام فارسی آنها وارد شده و با فشردن گزینه Apply ، تغییرات ذخیره می‌شود. (شکل ۵-۱۱)

---

[۱۰۳] CPU Usage



شکل ۵-۱۱ – پنل گزارش فعالیت CPU

۵-۱-۳- گزارش درگیری حافظه[۱۰۴]

این گزارش میزان مصرف حافظه در بازه‌های زمانی مختلف را نشان داده و شاخص‌های کمینه، بیشینه و میانگین استفاده از RAM را ثبت می‌کند. از طریق این گزارش می‌توان وضعیت بهینه بودن تخصیص حافظه، وجود نشت حافظه (Memory Leak) و یا نیاز به ارتقاء سخت‌افزاری را تحلیل نمود.

مرحله ۶: ایجاد پنل گزارش درگیری RAM

با ایجاد پنل جدید در همان داشبورد، در قسمت Query عبارت زیر وارد می‌شود:

```
100 * (1 - (node_memory_MemAvailable_bytes{instance=~"$instance"} /
node_memory_MemTotal_bytes{instance=~"$instance"}))
```

از بخش Visualization نوع نمایش را Stat یا Table (در صورت ترجیح عددی) انتخاب و سپس در قسمت Panel title عنوان دلخواه همانند «RAM Usage (%)» وارد می‌شود. در Standard options → Unit، گزینه‌ی percent (0-100) و از Standard options → Decimals عدد ۲ انتخاب می‌شود.

---

[۱۰۴] RAM Usage

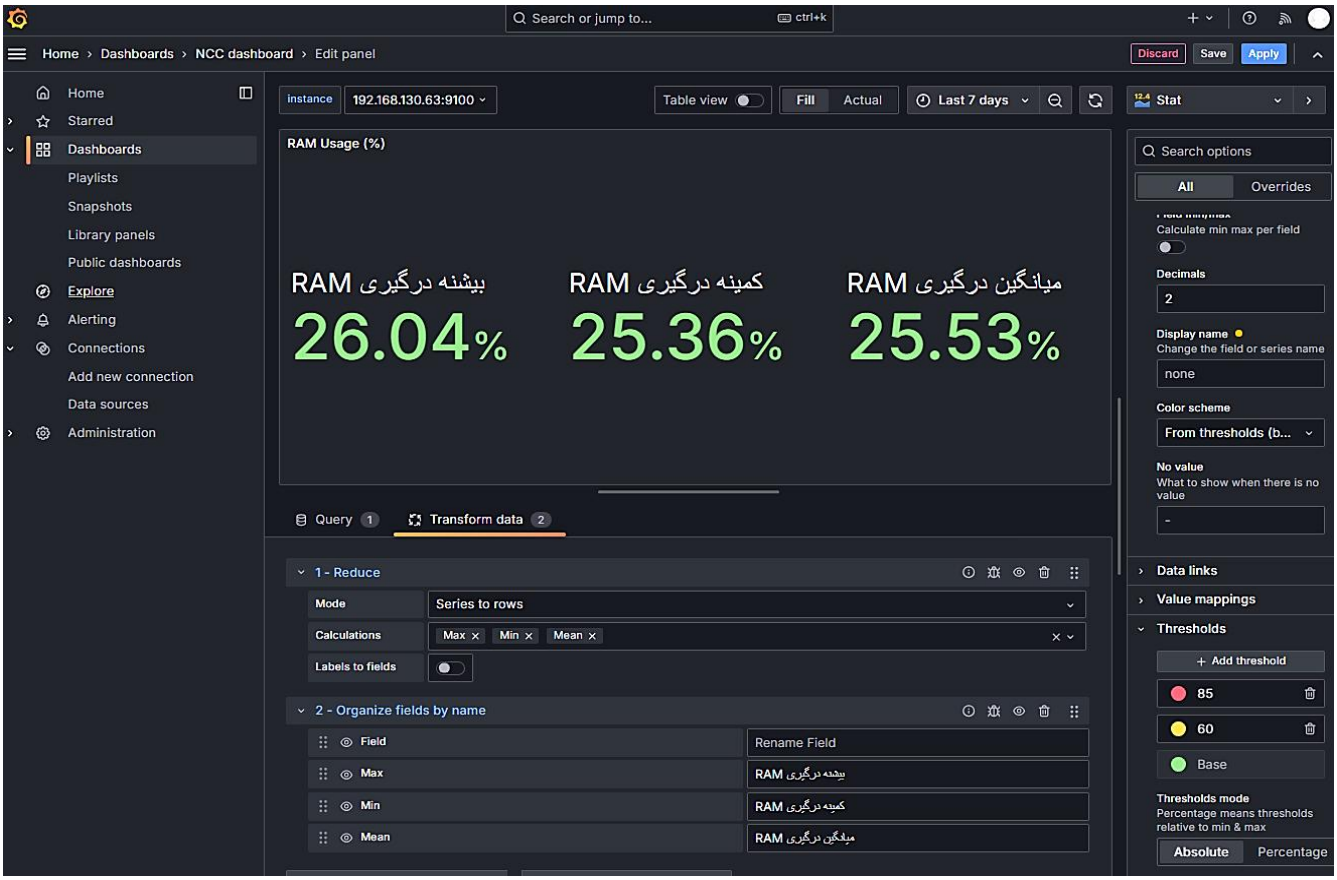
در قدم بعدی، در بخش پایینی پنل در تب Transform data با کلیک بر روی Add transformation، گزینه‌ی Reduce انتخاب می‌شود. در قسمت Calculation نیز، گزینه‌های Min و Max و Mean فعال می‌شوند. در بخش Mode of calculation، با انتخاب گزینه‌ی Series to rows هر مقدار در یک ستون جداگانه نمایش داده خواهد شد. برای قالب‌بندی نمایش، در بخش Thresholds، به‌عنوان نمونه، رنگ‌های سبز، زرد و قرمز برای محدوده‌های زیر به این صورت تنظیم می‌شوند:

تا ۶۰٪: سبز

۶۰ تا ۸۵٪: زرد

بالاتر از ۸۵٪: قرمز

همانند مرحله قبل نام ستون‌ها فارسی شده و تغییرات ذخیره می‌شوند. (شکل ۵-۱۲)



شکل ۵-۱۲ – پنل گزارش درگیری RAM

۵-۱-۴- گزارش وضعیت دیسک[۱۰۵]

این گزارش شامل بررسی فضای مصرفی دیسک و زمان پاسخ‌دهی دیسک است. هدف از این گزارش، پیش‌بینی کمبود فضای ذخیره‌سازی، شناسایی بخش‌هایی است که در آستانه پر شدن یا کاهش کارایی قرار دارند، و برنامه‌ریزی برای نگهداری، پاک‌سازی یا

[۱۰۵] Disk Utilization

انتقال داده‌هاست. همچنین گزارش زمان پاسخ‌دهی دیسک می‌تواند در زمان بروز کندی در سرویس یا ابزارهایی مانند ژئوسرور یا تامکت، به تشخیص نقش احتمالی دیسک در ایجاد مشکل کمک کند.

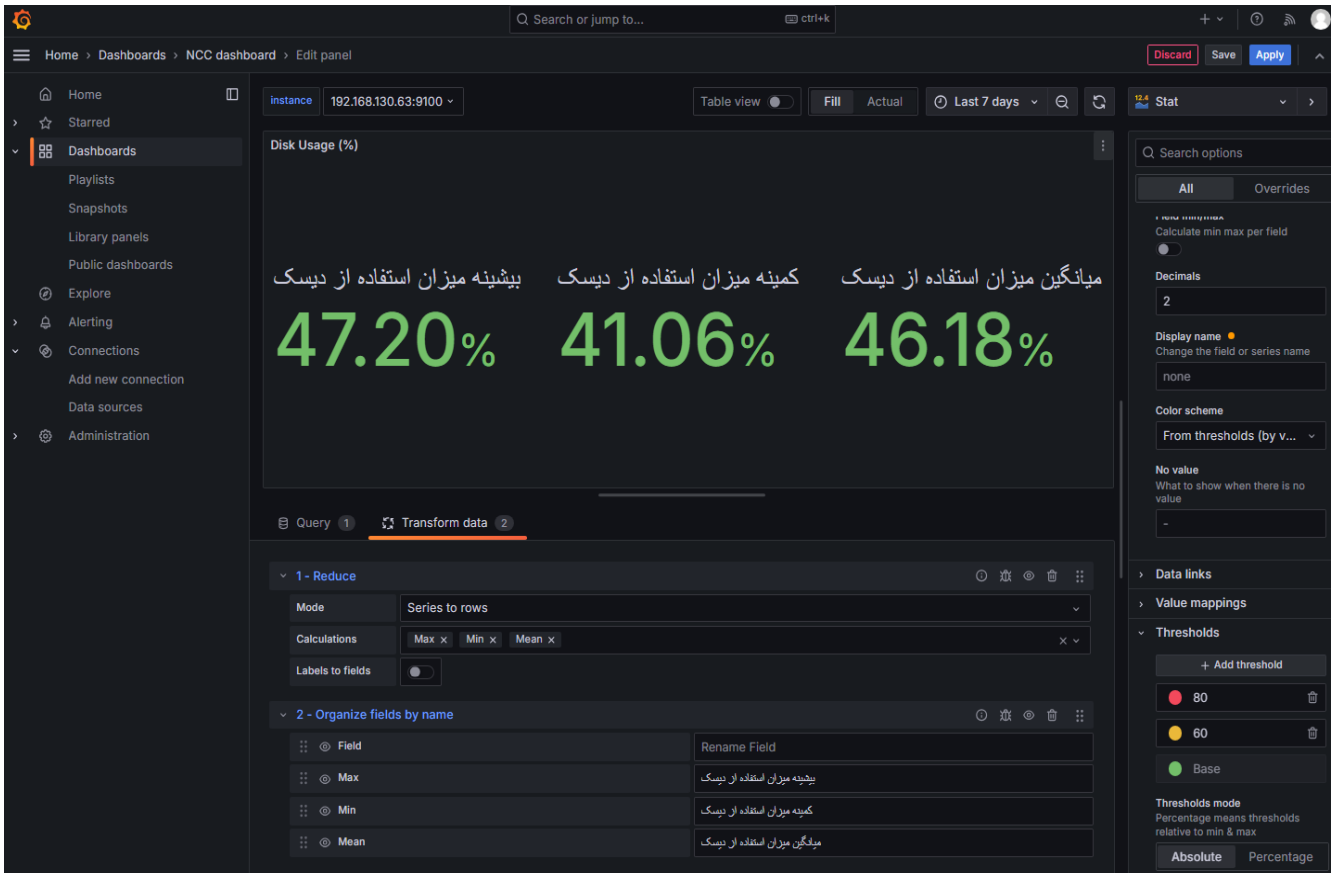
مرحله ۷: ایجاد پنل گزارش وضعیت استفاده از Disk

در همان داشبورد پایش، با ایجاد پنل جدید، در قسمت Query عبارت زیر وارد می‌شود:

```
100 * (1 - (node_filesystem_avail_bytes{mountpoint="/",fstype!="rootfs",instance=~"$instance"} /
node_filesystem_size_bytes{mountpoint="/",fstype!="rootfs",instance=~"$instance"}))
```

در بخش Visualization، گزینه‌ی Stat را انتخاب و در عنوان پنل مقدار دلخواهی مانند «Disk Usage (%)» قرار داده و در قسمت Standard options → Unit، گزینه‌ی percent (0-100) و در قسمت Standard options → Decimals عدد ۲ انتخاب می‌شود.

همانند مراحل قبل، در بخش پایینی پنل، در تب Transform data روی Add transformation کلیک و گزینه‌ی Reduce انتخاب می‌شود. در قسمت Calculation، گزینه‌های Min و Max و Mean را فعال می‌شوند. در بخش Mode of calculation، با انتخاب گزینه‌ی Series to rows قالب‌بندی نمایش تنظیم می‌شود. در پایان، همانند مرحله قبل، نام ستون‌ها را فارسی و تغییرات ذخیره می‌شوند. (شکل ۵-۱۳)



شکل ۵-۱۳ – پنل گزارش فضای استفاده شده از دیسک

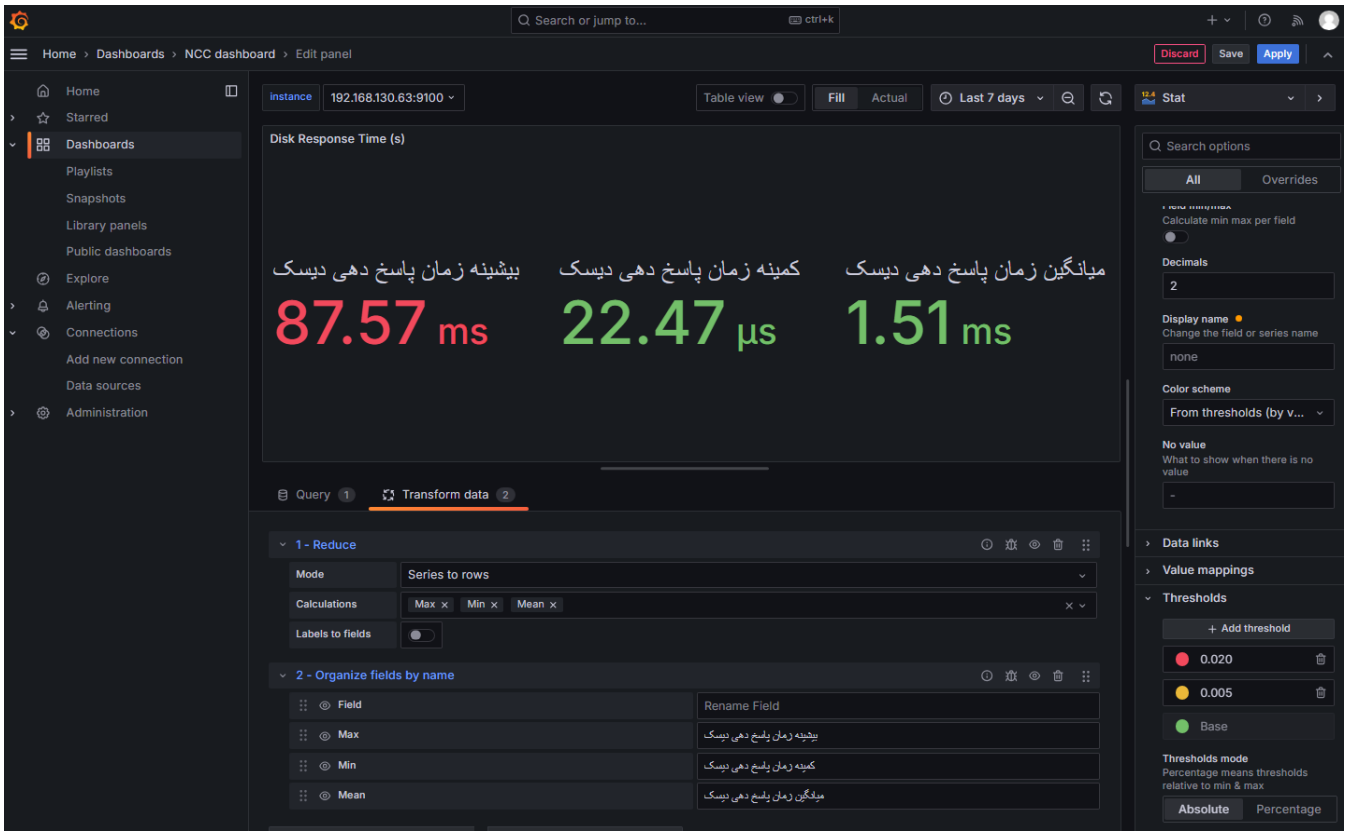
مرحله ۸: ایجاد پنل گزارش وضعیت پاسخ‌دهی Disk

این متریک نرخ زمانی را نشان می‌دهد که دستگاه ذخیره‌سازی در حال انجام عملیات I/O بوده است؛ مقدار آن هرچه بیشتر باشد، پاسخ‌دهی دیسک کندتر است. برای گزارش‌دهی پنل جدیدی به شرح ذیل در همان داشبورد ایجاد می‌شود:

```
rate(node_disk_io_time_seconds_total{instance=~"$instance"}[5m])
```

در بخش Visualization نوع نمایش را Stat انتخاب کرده و در قسمت Panel title، عنوان «Disk Response Time (s)» را نوشته و در Standard options → Unit، گزینه Seconds و در قسمت Decimals عدد ۲ انتخاب می‌شود.

همانند سایر مراحل، در بخش پایینی پنل در تب Transform data با کلیک بر روی Add transformation گزینه‌ی Reduce انتخاب می‌شود. در قسمت Calculation، گزینه‌های Min و Max و Mean را فعال کرده و سپس قالب‌بندی نمایش و فارسی‌سازی سرستون‌ها انجام می‌شود. (شکل ۵-۱۴)



شکل ۵-۱۴ – پنل گزارش زمان پاسخ‌دهی دیسک

۵-۱-۵- گزارش شبکه[۱۰۶]

در این بخش وضعیت ترافیک ورودی و خروجی هر سرور پایش می‌شود. این شاخص برای بررسی کیفیت ارتباط میان سرورها، بار شبکه و شناسایی گلوگاه‌های احتمالی در انتقال داده‌ها اهمیت بالایی دارند.

[۱۰۶] Network Reports

مرحله ۹: ایجاد پنل گزارش ترافیک شبکه

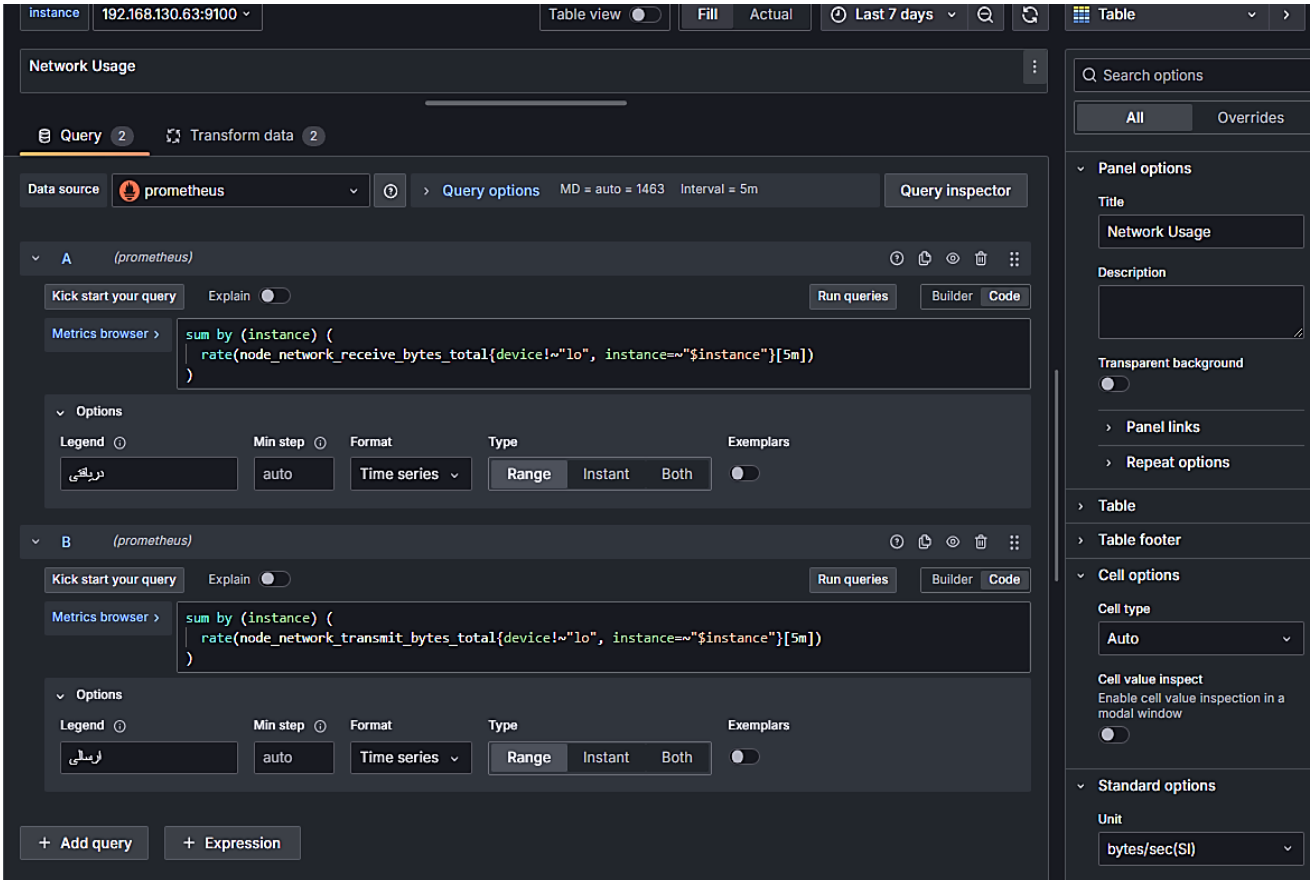
گزارش شامل سه شاخص کمینه، بیشینه و میانگین میزان استفاده از پهنای باند است. پنلی جدید ایجاد کرده و Visualization به صورت Table تنظیم می‌شود. در قسمت Panle Tiltle عبارت دلخواهی مانند «Network Usage» وارد کرده و قسمت Standard options → Unit، گزینه‌ی bytes/sec(SI) و در قسمت Decimals عدد ۲ انتخاب می‌شود. در قسمت Query برای ترافیک دریافتی کد زیر وارد می‌شود:

```
sum by (instance) (
  rate(node_network_receive_bytes_total{device!~"lo", instance=~"$instance"}[5m])
)
```

در همان قسمت بر روی Add Query کلیک کرده و برای گزارش ترافیک ارسالی کد زیر وارد می‌شود:

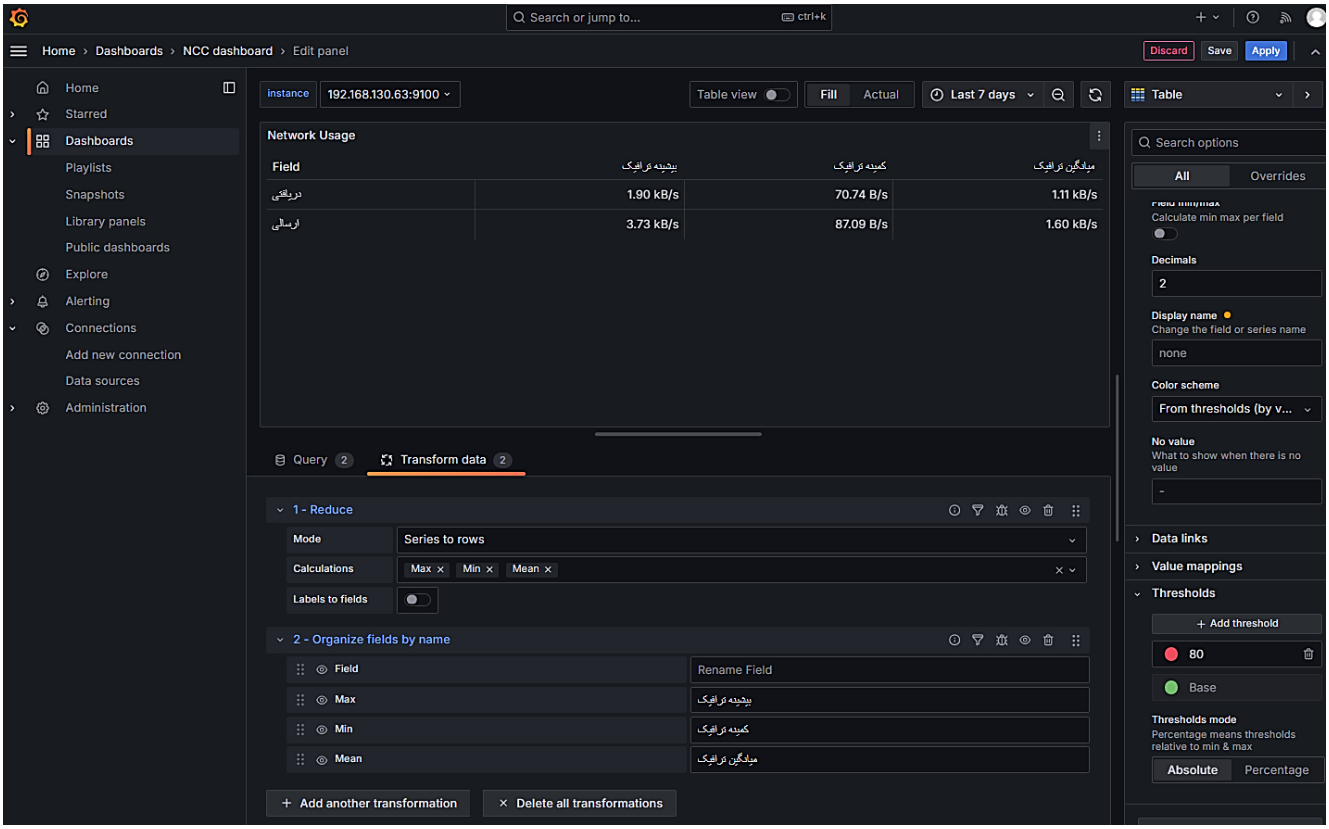
```
sum by (instance) (
  rate(node_network_transmit_bytes_total{device!~"lo", instance=~"$instance"}[5m])
)
```

در بخش پایینی هر کد کوئری، بر روی گزینه options کلیک کرده و در باکس Legend حالت Custom انتخاب و در کوئری اول عبارت «ترافیک دریافتی» و در کوئری دوم عبارت «ترافیک ارسالی» تایپ می‌شوند. (شکل ۵-۱۵)



شکل ۵-۱۵ – بخش اول تنظیمات پنل گزارش ترافیک شبکه

در ادامه، در بخش پایینی پنل در تب Transform data روی Add transformation کلیک و گزینه‌ی Reduce انتخاب می‌شود. در قسمت Calculation، گزینه‌های Min و Max و Mean را فعال نموده و در بخش Mode of calculation، گزینه‌ی Series to rows انتخاب می‌شود. در پایان همانند مراحل قبل با استفاده از Organize fields by name نام ستون‌ها فارسی و در پایان تغییرات ذخیره می‌شوند. (شکل ۵-۱۶)



شکل ۵-۱۶ – بخش دوم تنظیمات پنل گزارش ترافیک شبکه

۵-۱-۶- گزارش دسترس‌پذیری تامکت

این گزارش نشان‌دهنده‌ی وضعیت فعال بودن تامکت و زمان در دسترس بودن آن در بازه‌های زمانی مختلف است. از آن‌جا که اجرای ژئوسرور وابسته به بستر تامکت است، پایش این شاخص از اهمیت ویژه‌ای برخوردار است. هرگونه توقف یا کاهش زمان فعالیت تامکت به‌طور مستقیم بر سرویس‌های مکانی سازمان (مانند WMS، WFS و WCS) تأثیر گذاشته و باید به‌صورت دقیق بررسی شود.

مرحله ۱۰: ایجاد پنل گزارش قطعی‌های تامکت

هدف این گزارش فهرست زمان‌هایی که سیستم روشن بوده اما تامکت در حال سرویس‌دهی نبوده، همراه با زمان شروع و مدت قطعی است. به‌این‌منظور، در همان داشبورد پایش، بر روی منوی پنل استخراج مقاطع قطعی سرور (علامت : گوشه سمت راست) (مرحله ۴:) کلیک کرده و زیرگزینه Duplicate از گزینه More انتخاب می‌شود تا نیازی به اجرای مجدد و تنظیمات مشابه نباشد. در

پنل جدید وارد محیط ادیت شده و نام پنل را به عبارت «Tomcat Downtime Report» تغییر داده و در قسمت Query:A کد زیر و بعد با دکمه Add query کوئری B ایجاد کرده و کد دوم وارد می‌شود و سایر تنظیمات به‌قوت خود باقی است:

```
tomcat_down_event{job="server1_tomcat"}  # Query A
tomcat_up_event{job="server1_tomcat"}    # Query B
```

با اجرای کوئری‌ها گزارش مدنظر از مقاطع قطعی تامکت در حالی که سرور روشن و در دسترس است، حاصل می‌گردد.

۵-۲- گزارش‌های دریافتی از GeoHealthCheck

GeoHealthCheck با ارسال دوره‌ای درخواست‌های استاندارد OGC به سرویس‌های مکانی، وضعیت دسترس‌پذیری، زمان پاسخ و نتیجه اجرای هر درخواست را بررسی و ثبت می‌کند. گزارش‌های تولیدشده توسط این سامانه، امکان ارزیابی عملکرد عملیاتی سرویس‌های مکانی و شناسایی اختلالات احتمالی را فراهم می‌سازند.

گزارش‌های مورد نیاز از GeoHealthCheck در این مستند، شامل موارد زیر هستند:

۵-۲-۱- گزارش دسترس‌پذیری سرویس‌های مکانی

در این گزارش، وضعیت دسترس‌پذیری و پاسخ‌گویی سرویس‌های مکانی سازمان (از جمله WMS، WFS و WCS) پایش می‌شود. نتایج این گزارش نشان می‌دهد که هر سرویس در بازه زمانی موردنظر تا چه اندازه در دسترس بوده و درخواست‌های پایش را با موفقیت پاسخ داده است. کاهش این شاخص می‌تواند ناشی از اختلال در سرویس، توقف بستر اجرای آن یا مشکلات ارتباطی باشد.

۵-۲-۲- گزارش پاسخ‌دهی به درخواست GetCapabilities

این گزارش نتیجه اجرای Probe از نوع GetCapabilities را نمایش می‌دهد. در این آزمون، درخواست استاندارد GetCapabilities به سرویس ارسال شده و در دسترس بودن سرویس، معتبر بودن پاسخ و امکان دریافت اطلاعات توصیفی آن، از جمله لایه‌های منتشرشده و قابلیت‌های سرویس، بررسی می‌شود. این گزارش برای ارزیابی صحت پاسخ‌گویی سرویس به درخواست‌های پایه استاندارد OGC مورد استفاده قرار می‌گیرد.

۵-۲-۳- گزارش پاسخ‌دهی به درخواست GetResource یا (GetFeature / GetMap)

در این گزارش، Probe از نوع GetResource با اجرای درخواست‌های عملیاتی مانند GetMap یا GetFeature، عملکرد واقعی سرویس در ارائه داده‌های مکانی را ارزیابی می‌کند. نتایج این آزمون امکان بررسی موفقیت اجرای درخواست‌ها، زمان پاسخ و شناسایی خطاهای احتمالی در فرآیند ارائه داده‌های مکانی را فراهم می‌سازد.

پیوست ۱: متن فایل pavement.py

```
# ================================================================
#
# Authors: Tom Kralidis <tomkralidis@gmail.com>
#
# Copyright (c) 2023 Tom Kralidis
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of this software and associated documentation
# files (the "Software"), to deal in the Software without
# restriction, including without limitation the rights to use,
# copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the
# Software is furnished to do so, subject to the following
# conditions:
#
# The above copyright notice and this permission notice shall be
# included in all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
# EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
# OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
# NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
# WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
# OTHER DEALINGS IN THE SOFTWARE.
#
# ================================================================

import codecs
import glob
import os
import shutil
import tempfile
from io import BytesIO
from urllib.request import urlopen
import zipfile

from paver.easy import (Bunch, call_task, cmdopts, info, options,
                        path, pushd, sh, task)

BASEDIR = os.path.abspath(os.path.dirname(__file__))

options(
    base=Bunch(
        home=path(BASEDIR),
        docs=path('%s/docs' % BASEDIR),
        instance=path('%s/instance' % BASEDIR),
        pot=path('%s/GeoHealthCheck/translations/en/LC_MESSAGES/messages.po' % BASEDIR),
        static_docs=path('%s/GeoHealthCheck/static/docs' % BASEDIR),
        static_lib=path('%s/GeoHealthCheck/static/lib' % BASEDIR),
        tmp=path(tempfile.mkdtemp()),
```

```
        translations=path('%s/GeoHealthCheck/translations' % BASEDIR),
        assets=path('%s/instance/assets' % BASEDIR)  # مسیر فایل‌های محلی
    ),
)

@task
def setup():
    config_file = options.base.home / 'GeoHealthCheck/config_main.py'
    config_site = options.base.instance / 'config_site.py'

    # setup dirs
    if not os.path.exists(options.base.static_lib):
        options.base.static_lib.mkdir()
    if not os.path.exists(options.base.instance):
        options.base.instance.mkdir()
        data_dir = options.base.instance / 'data'
        data_dir.mkdir()
        # data_dir.chmod(0777) gives failure on Python 2.7 Paver 1.2.1
        os.chmod(path(data_dir), 0o777)
        # setup config
        config_file.copy(config_site)

    # setup deps
    sh('pip install -r requirements.txt')

    # بارگذاری فایل‌ها از مسیر محلی به static/lib
    assets = options.base.assets

    # startbootstrap-sb-admin-2
    sb_zip = assets / 'startbootstrap-sb-admin-2-3.3.7-1.zip'
    with zipfile.ZipFile(sb_zip) as zipfile_obj:
        zipfile_obj.extractall(options.base.static_lib)

    for folder in ['dist', 'vendor']:
        src = path(options.base.static_lib / 'startbootstrap-sb-admin-2-3.3.7-1' / folder)
        dst = path(options.base.static_lib / folder)
        if not os.path.exists(dst):
            src.move(dst)
    shutil.rmtree(path(options.base.static_lib / 'startbootstrap-sb-admin-2-3.3.7-1'))

    # select2
    select2_zip = assets / 'select2-4.0.3.zip'
    with zipfile.ZipFile(select2_zip) as zipfile_obj:
        zipfile_obj.extractall(options.base.static_lib)
    dirname = glob.glob(options.base.static_lib / 'select2-*')[0]
    dstdir = ''.join(dirname.rsplit('-', 1)[:-1])
    try:
        os.rename(dirname, dstdir)
    except OSError:
        shutil.rmtree(dstdir)
        os.rename(dirname, dstdir)

    # leaflet
    leaflet_js = assets / 'leaflet.min.js'
    if os.path.exists(leaflet_js):
        shutil.copy(leaflet_js, options.base.static_lib / 'leaflet.min.js')
    else:
```

```
        info('Leaflet asset not found in assets')



    # html5shiv
    shutil.copy(assets / 'html5shiv.min.js', options.base.static_lib / 'html5shiv.min.js')

    # respond.min.js
    shutil.copy(assets / 'respond.min.js', options.base.static_lib / 'respond.min.js')

    # build i18n .mo files
    call_task('compile_translations')

    # build local docs
    call_task('refresh_docs')

    info('GeoHealthCheck is now built. Edit settings in %s' % config_site)
    info('before deploying the application. Alternatively, you can start a')
    info('development instance with "python GeoHealthCheck/app.py"')


@task
def create_secret_key():
    """create secret key for SECRET_KEY in instance/config_site.py"""
    info('Secret key: \"%s\" % codecs.encode(os.urandom(24), 'hex').decode())
    info('Copy/paste this key to set the SECRET_KEY')
    info('value in instance/config_site.py')


@task
@cmdopts([
    ('email=', 'e', 'email'),
    ('username=', 'u', 'username'),
    ('password=', 'p', 'password')
])
def create(options):
    """create database objects and superuser account"""

    args = ''
    username = options.get('username', None)
    password = options.get('password', None)
    email = options.get('email', None)

    if all([username, password, email]):
        args = '%s %s %s' % (username, password, email)
    sh('python3 %s create %s' % (path('GeoHealthCheck/models.py'), args))


@task
@cmdopts([
    ('password=', 'p', 'password')
])
def create_hash(options):
    """
    Create hash, mainly for passwords.
    Usage: paver create_hash -p mypass
    """
```

```
    import sys
    sys.path.insert(0, BASEDIR + '/GeoHealthCheck')
    from util import create_hash
    token = create_hash(options.get('password', None))
    info('Copy/paste the entire token below for example to set password')
    info(token)


@task
def upgrade():
    """upgrade database if changed; be sure to backup first!"""

    info('Upgrading database...')
    with pushd(path('%s/GeoHealthCheck' % BASEDIR)):
        sh('python3 manage.py db upgrade')


@task
def create_wsgi():
    """create WSGI wrapper and Apache2 configuration"""
    wsgi_script = '%s%sGeoHealthCheck.wsgi' % (options.base.instance, os.sep)
    with open(wsgi_script, 'w') as ff:
        ff.write('import sys\n')
        ff.write('sys.path.insert(0, \'%s\')\n' % BASEDIR)
        ff.write('from GeoHealthCheck.app import APP as application')

    wsgi_conf = '%s%sGeoHealthCheck.conf' % (options.base.instance, os.sep)
    with open(wsgi_conf, 'w') as ff:
        ff.write('WSGIScriptAlias / %s%sGeoHealthCheck.wsgi\n' %
                 (options.base.instance, os.sep))
        ff.write('<Directory %s%s>\n' % (BASEDIR, os.sep))
        ff.write('Order deny,allow\n')
        ff.write('Allow from all\n')
        ff.write('</Directory>')


@task
def refresh_docs():
    """Build sphinx docs from scratch"""

    make = sphinx_make()

    if os.path.exists(options.base.static_docs):
        shutil.rmtree(options.base.static_docs)

    with pushd(options.base.docs):
        sh('%s clean' % make)
        sh('%s html' % make)
        source_html_dir = path('%s/docs/_build/html' % BASEDIR)
        source_html_dir.copytree(options.base.static_docs)


@task
def clean():
    """clean environment"""

    if os.path.exists(options.base.static_lib):
```

```
        shutil.rmtree(options.base.static_lib)
    if os.path.exists(options.base.tmp):
        shutil.rmtree(options.base.tmp)
    if os.path.exists(options.base.static_docs):
        shutil.rmtree(options.base.static_docs)


@task
def extract_translations():
    """extract translations wrapped in _() or gettext()"""

    pot_dir = path('GeoHealthCheck/translations/en/LC_MESSAGES')
    if not os.path.exists(pot_dir):
        pot_dir.makedirs()

    sh('pybabel extract -F babel.cfg -o %s GeoHealthCheck' % options.base.pot)


@task
@cmdopts([
    ('lang=', 'l', '2-letter language code'),
])
def add_language_catalogue(options):
    """adds new language profile"""

    lang = options.get('lang', None)

    if lang is None:
        raise RuntimeError('missing lang argument')

    sh('pybabel init -i %s -d %s -l %s' % (
       options.base.pot, options.base.translations, lang))


@task
def compile_translations():
    """build .mo files"""

    sh('pybabel compile -d %s' % options.base.translations)


@task
def update_translations():
    """update language strings"""

    call_task('extract_translations')
    sh('pybabel update -i %s -d %s' % (
        options.base.pot, options.base.translations))


@task
def runner_daemon():
    """Run the HealthCheck runner daemon scheduler"""
    sh('python3 %s' % path('GeoHealthCheck/scheduler.py'))


@task
def run_healthchecks():
```

```
    """Run all HealthChecks directly"""
    sh('python3 %s' % path('GeoHealthCheck/healthcheck.py'))


def sphinx_make():
    """return what command Sphinx is using for make"""

    if os.name == 'nt':
        return 'make.bat'
    return 'make'


@task
def run_tests():
    """Run all tests"""
    sh('python3 %s' % path('tests/run_tests.py'))
```



Islamic Republic of Iran
Plan and Budget Organization
National Cartographic Center

Technical Guide for Preliminary Monitoring of Geospatial Services and Their Underlying Infrastructure

No: 01

Last Edition: 10-12-2026

National Cartographic Center

www.ncc.gov.ir

Standard Department

https://ncc.gov.ir/استاندارد-ها

2026