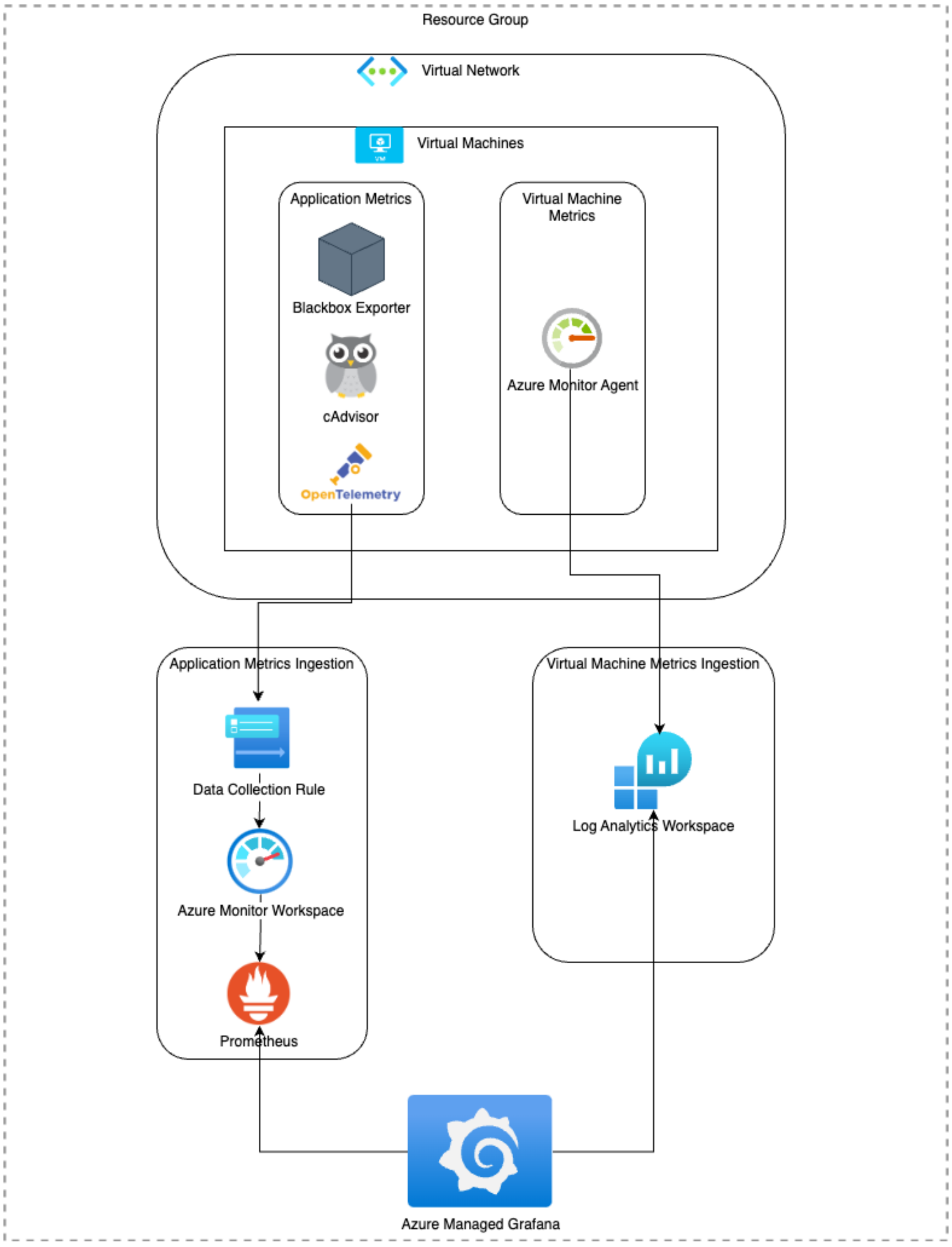
VM, endpoint & performance monitoring

Architecture

Diagram



Description

This architecture presents the collection of system & application metrics using Azure services such as Azure log Analytics & Azure Monitor workspaces as well as Azure managed Grafana. Grafana dashboards are used to visualize all the collected metrics.

For the system metrics, the diagram presents the collection of system metrics through Azure monitor agent which sends the metrics to Azure Log Analytics service.

As for the application metrics such as http status and ssl certificate validity, blackbox exporter is installed to collect the metrics and ingest it in the Opentelemetry collector. The Opentelemetry collector uses Azure Active Directory Authentication Proxy for managing the authentication token so that the metrics can be ingested in Azure Monitor Workspace through the existing Data collection Rule.

Deployment

Applications

Counter application is used as a demo application where the application endpoints have self-signed certificates. The application endpoints are monitored in terms of http status and SSL certificates validity.

PicoShare is a containerized application that allows you to share files easily. The application is used to measure the upload/download speeds of file sharing.

Infrastructure Provisioning using Terraform

The infrastructure components are provisioned using Terraform and the Terraform state is managed in Azure Blob Storage. This approach makes sure that the state is consistently saved and accessible for future updates.

Pre-requisites

Before provisioning ensure that the following tools are installed and properly configured:

1. Terraform: For provisioning the infrastructure
2. Azure CLI: To authenticate with Microsoft Azure
3. Ansible: For configuration management

Usage

1. Azure Authentication

It is necessary to login to the specific Azure account to allow terraform to interact with the resources. After executing the login command a browser window will open for authentication. After completion, the Azure CLI session is authenticated successfully.

```
az login
```

2. Customize terraform.tfvars file for Bootstraping

The bootstrap directory is used to create the necessary resources that will hold the Terraform state file. One should navigate to the bootstrap directory and customize the variables in the terraform.tfvars file with the correct values for the specific azure environment.

terraform.tfvars

```
tenant_id = "azure-tenant-id"
subscription_id = "azure-subscription-id"
storage_account_name = "azure-storage-account-name"
resource_group_name = "azure-resource-group-name"
resource_group_location = "azure-resource-group-location"
name_prefix = "name-prefix"
```

3. Initialize, Plan, and Apply the Bootstrap Resources

For running the Terraform commands, the workspace needs to be initialized and the Terraform State infrastructure needs to be created. For this, a bootstrap module has been created, which will create a Resource Group, Storage Account, Storage Container and a Management Lock on the Storage Account.

To run the bootstrap, navigate to **./infra/bootstrap** and update the terraform.tfvars respectively. After that, run terraform init and terraform apply.

```
terraform init -backend-config=backend.hcl
```

Once initialized, a plan could be generated to preview resources Terraform will create.

```
terraform plan
```

The plan should be reviewed, and if everything looks fine the plan could be applied by executing the terraform apply command to create the resources.

```
terraform apply
```

4. Initialize, Plan, and Apply the Compute & Monitoring Resources

Again we need to update the terraform.tfvars file and fill in the needed information. Note that resource_group_name & location, aswell as storage_account_name should be taken from the bootstrap module applied before.

Additionally, the backend.hcl file should be filled with the details.

Initialize the root module.

```
terraform init -backend-config=backend.hcl
```

Once initialized, a plan will be generated to preview resources Terraform will create.

```
terraform plan
```

The plan should be reviewed, and if everything looks fine, the plan could be applied by executing the terraform apply command to create the resources.

```
terraform apply
```

Configuration Management using Ansible

After the Infrastructure is provisioned using Terraform, Ansible is used to configure the VMs and deploy the necessary monitoring tools.

1. Gather VM IP Addresses

Terraform outputs the IP addresses of VMs after the deployment is successfully finished. These IP addresses could be gathered from the Terraform output or retrieved from the Azure portal.

2. Update/Create the Ansible Inventory File

The Ansible inventory file should be updated by adding the IP addresses of the VMs under the [all] group, the correct path to the SSH private key and the user are under [all:vars].

3. Update Variables

Under ./ansible/vars you will find a otel_vars.yml file where variables need to be set. The data_collection_rule_id can be obtained by navigating to the “Data Collection Rules” Resource in the Azure Portal and selecting the “*-azure-monitor" Rule and copying the “Immutable ID”.

The azure_monitor_ingestion_host can be obtained by navigating to the created Azure Monitor Workspace and copying the host portion of the Metrics Ingestion Endpoint (e.g. https://azure-monitor.westeurope-1.metrics.ingest.monitor.azure.com).

The user_identity_id can be obtained by running “terraform output” on the root module.

4. Run the Ansible Playbook

Navigate to the directory containing the Ansible playbook and run it with the inventory file. By executing the ansible-playbook command, ansible will configure the VMs as defined in the playbook.yml, using the IPs and key provided in the inventory file.

```
ansible-playbook -i inventory.ini playbook.yml
```

Monitoring

Data Sources

Two datasources are configured to collect the metrics from the VMs.

Azure Monitor

System metrics are collected through Azure Monitor agent deployed on the VMs. The collected metrics include CPU, RAM and Disk usage.

Prometheus (Azure Monitor workspace)

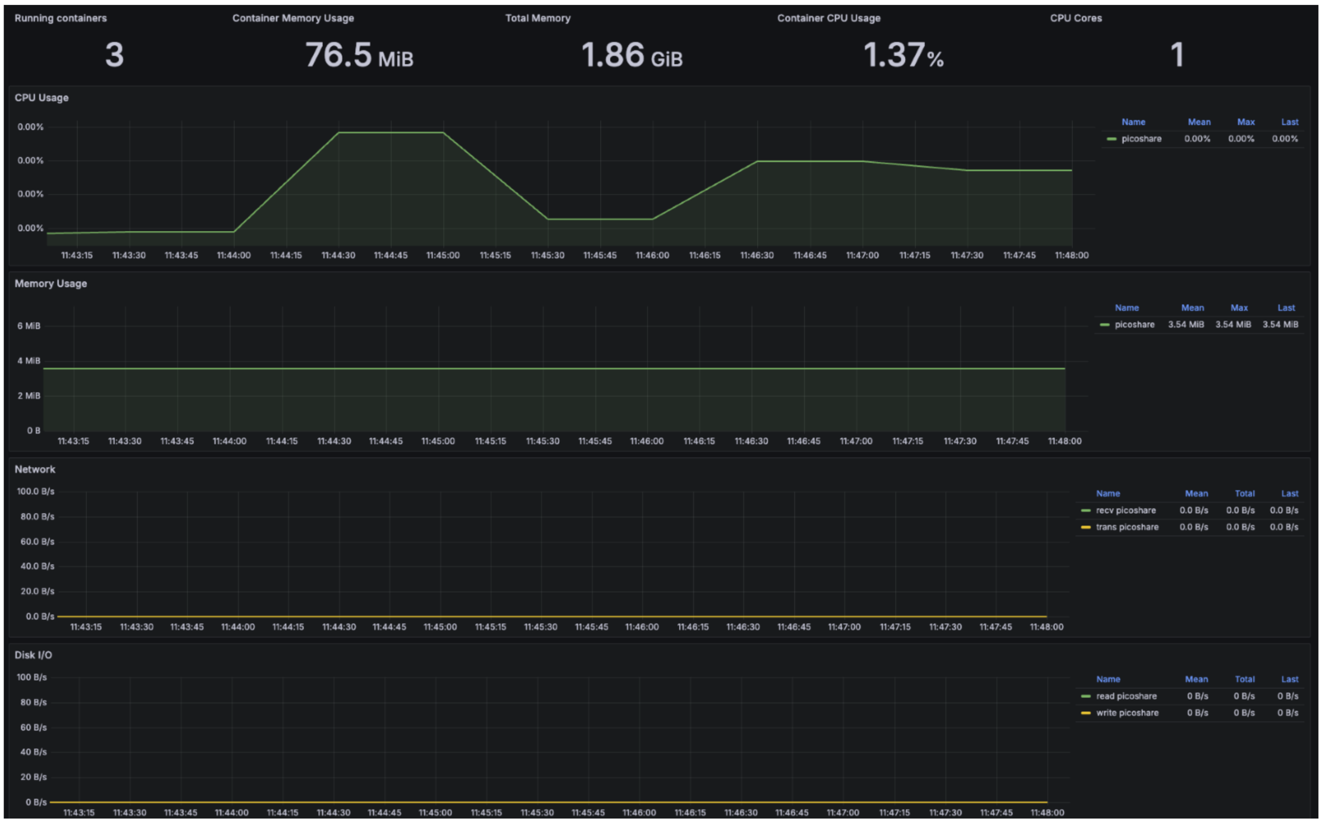
Application metrics are collected using blackbox and cadvisor exporters. Blackbox exporter allows blackbox probing of endpoints over HTTP, HTTPS. Metrics such as http/s status and ssl certificate validity are collected through blackbox exporter.

cAdvisor (Container Advisor) provides container users an understanding of the resource usage and performance characteristics of their running containers. It is a running daemon that collects, aggregates, processes, and exports information about running containers.

Dashboards

In the repository you will find 2 JSON Dashboard files which can be imported into Grafana. The docker_monitor dashboards provides container insights, especially network traffic. The Blackbox Dashboard will provide detailed insights into the monitored Clicker applications SSL Validity.

Upload & Download Speed of Filetransfer service



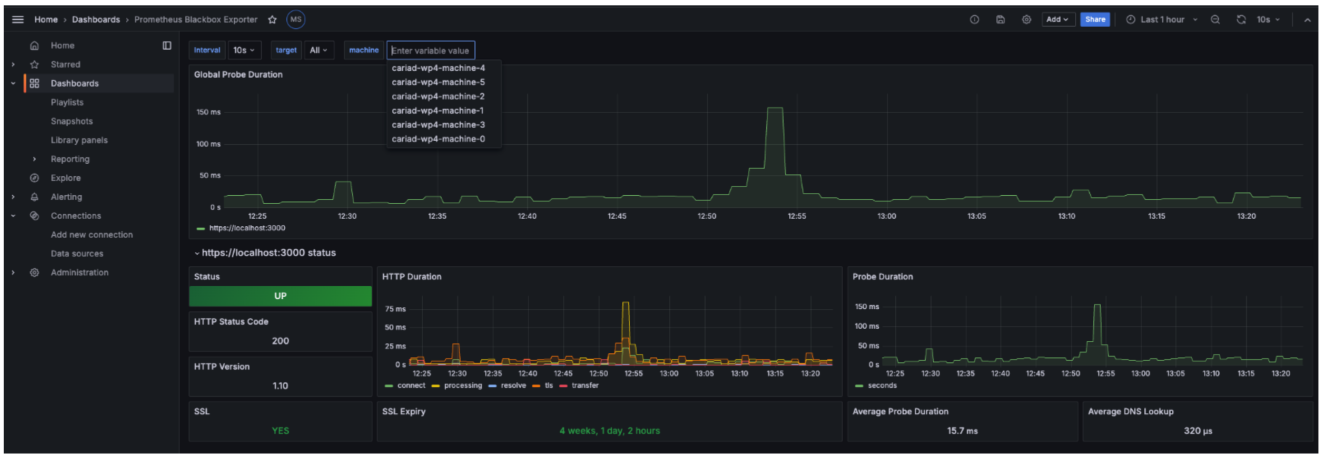
Name	Mean	Max	Last
picoshare	0.00%	0.00%	0.00%

Name	Mean	Max	Last
picoshare	3.54 MiB	3.54 MiB	3.54 MiB

Name	Mean	Total	Last
recv picoshare	0.0 B/s	0.0 B/s	0.0 B/s
trans picoshare	0.0 B/s	0.0 B/s	0.0 B/s

Name	Mean	Total	Last
read picoshare	0 B/s	0 B/s	0 B/s
write picoshare	0 B/s	0 B/s	0 B/s

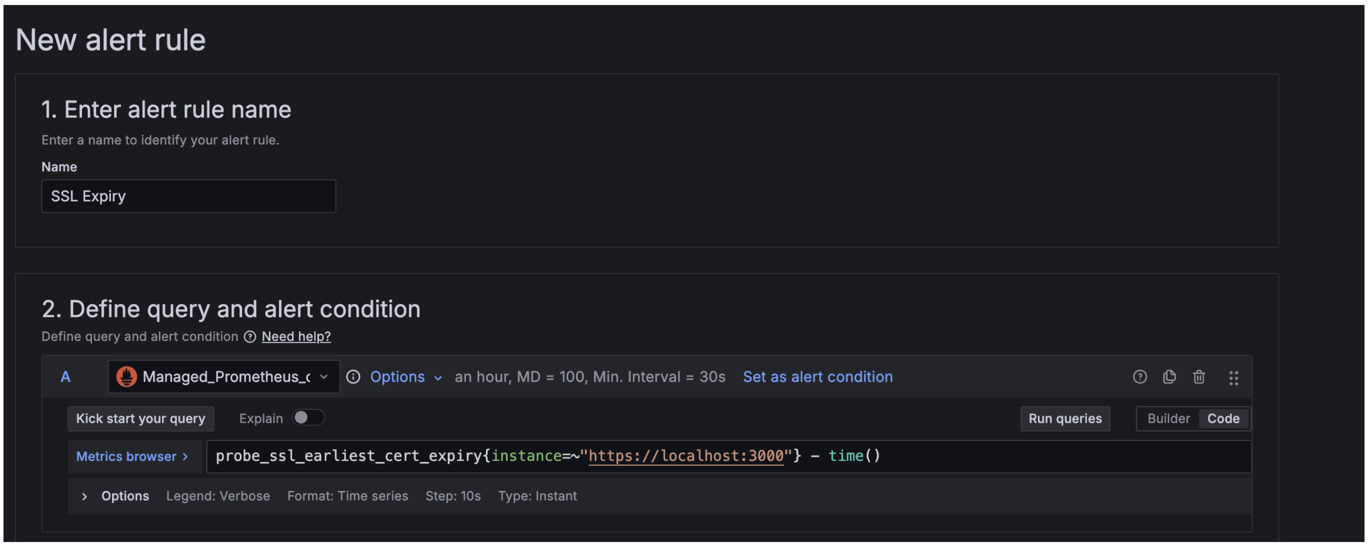
SSL Certificate Validity / HTTP Endpoint Monitoring



Status: UP
HTTP Status Code: 200
HTTP Version: 1.10
SSL: YES
SSL Expiry: 4 weeks, 1 day, 2 hours
Average Probe Duration: 15.7 ms
Average DNS Lookup: 320 µs

Alerting

Alerts can be created from Grafana based on Prometheus Metrics or already created dashboard charts. Following you will find an example configuration of an Alert for SSL Certificates that will expire in less than 1 week: **Alert Query**


New alert rule

1. Enter alert rule name

Enter a name to identify your alert rule.

Name: SSL Expiry

2. Define query and alert condition

Define query and alert condition Need help?

A Managed_Prometheus_ Options an hour, MD = 100, Min. Interval = 30s Set as alert condition

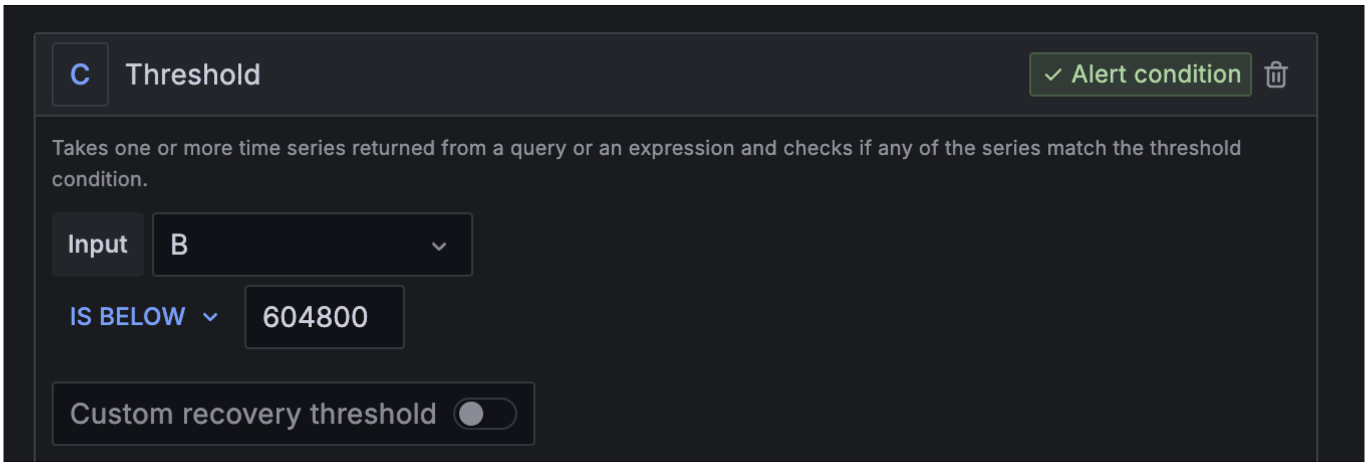
Kick start your query Explain Run queries Builder Code

Metrics browser

```
probe_ssl_earliest_cert_expiry{instance=~"https://localhost:3000"} - time()
```

Options Legend: Verbose Format: Time series Step: 10s Type: Instant

Alert configuration (1 week)


C Threshold ✓ Alert condition

Takes one or more time series returned from a query or an expression and checks if any of the series match the threshold condition.

Input: B

IS BELOW 604800

Custom recovery threshold