# Applications

- Loki, Alloy, and Prometheus Monitoring for Kubernetes

# Loki, Alloy, and Prometheus Monitoring for Kubernetes

This guide outlines the process for installing Loki, Alloy, and Prometheus on a Kubernetes cluster. This setup provides comprehensive logging and monitoring capabilities.

# Prerequisites

- A running Kubernetes cluster.
- Helm v3 or later.
- kubectl configured to interact with your cluster.

# 1. Prometheus Installation

Install Prometheus using Helm, ensuring the `server.extraArgs` argument is correctly set to allow Alloy to send data.

```
helm install prometheus-main prometheus-community/prometheus \
    --version 28.9.0 \
    --namespace logging \
    --set kube-state-metrics.enabled=false \
    --set prometheus-node-exporter.enabled=false \
    --set prometheus-pushgateway.enabled=false \
    --set "server.extraArgs.web.enable-remote-write-receiver="
```

# 2. Adding Custom Scrape Targets (Optional)

To scrape targets outside the Kubernetes cluster, edit the Prometheus ConfigMap.

```
scrape_configs:
  - job_name: 'pve'
    scrape_interval: 15s
    static_configs:
      - targets: ['192.168.2.250:9100']
    labels:
      instance: 'pve'
```

# 3. Loki Installation

Install Loki using Helm.

```
helm install loki-main grafana/loki -f loki-values.yaml --namespace logging
```

The values are a bit in-depth. These set the local filesystem for storage instead of S3, adds a 7-day retention period on logs, and lowers the requests as the default install is asking for 8GB per pod!

```
#loki-values.yaml
loki:
  storage:
    type: 'filesystem'
  auth_enabled: false
  commonConfig:
    replication_factor: 1
    # This explicitly routes all storage requests to the local disk
    path_prefix: /var/loki
    storage:
      filesystem:
        chunks_directory: /var/loki/chunks
        rules_directory: /var/loki/rules
  limits_config:
    retention_period: 168h  # <--- THIS deletes logs older than 7 days
    allow_deletes: true
  schemaConfig:
    configs:
      - from: "2024-04-01"
        store: tsdb
        object_store: filesystem
```

```
      schema: v13
      index:
        prefix: loki_index_
        period: 24h

  # Ensures the pod has a disk to write to
  storageConfig:
    filesystem:
      directory: /var/loki/chunks

  pattern_ingester:
      enabled: true
  limits_config:
    allow_structured_metadata: true
    volume_enabled: true
  ruler:
    enable_api: true

compactor:
  retention_enabled: true
  working_directory: /var/loki/retention
  delete_request_store: filesystem # <--- MUST match your storage type
  retention_delete_delay: 2h
  compaction_interval: 10m

minio:
  enabled: false

deploymentMode: SingleBinary

chunksCache:
  allocatedMemory: 512
  resources:
    requests:
      cpu: "10m"
      memory: "50Mi"
    limits:
      cpu: "50m"
      memory: "100Mi"
```

```
resultsCache:
  enabled: true
  allocatedMemory: 50
  resources:
    requests:
      cpu: "10m"
      memory: "50Mi"
    limits:
      cpu: "50m"
      memory: "100Mi"

singleBinary:
  replicas: 1
  persistence:
    enabled: true
    size: 10Gi
  resources:
    requests:
      cpu: "50m"      # Increase if logs are delayed
      memory: "50Mi"  # Increase if pod is OOMKilled

# Zero out replica counts of other deployment modes
backend:
  replicas: 0
read:
  replicas: 0
write:
  replicas: 0

ingester:
  replicas: 0
querier:
  replicas: 0
queryFrontend:
  replicas: 0
queryScheduler:
  replicas: 0
distributor:
```

```
  replicas: 0
compactor:
  replicas: 0
indexGateway:
  replicas: 0
bloomCompactor:
  replicas: 0
bloomGateway:
  replicas: 0
```

# 4. Alloy Installation

Install Alloy using Helm. Use --install if it's a new installation, or helm upgrade to update.

```
helm upgrade --install alloy grafana/alloy -n logging
```

We need to recreate the configmap to correctly label the pods and containers so we can search easily in grafana.

1. Delete the old configmap `kubectl delete cm -n logging alloy`
2. Apply the new configmap `kubectl apply -f alloy-cm.yaml`
3. Roll out new daemonset `kubectl rollout restart -n logging daemonset alloy`

```
#alloy-cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: alloy
  namespace: logging
data:
  config.alloy: |
    // Discover Kubernetes pods
    discovery.kubernetes "pods" {
      role = "pod"
    }

    // Relabel to add pod metadata as labels
    discovery.relabel "pod_logs" {
```

```
targets = discovery.kubernetes.pods.targets

// Add namespace
rule {
  source_labels = ["__meta_kubernetes_namespace"]
  target_label  = "namespace"
}

// Add pod name
rule {
  source_labels = ["__meta_kubernetes_pod_name"]
  target_label  = "pod"
}

// Add container name
rule {
  source_labels = ["__meta_kubernetes_pod_container_name"]
  target_label  = "container"
}

// Add node name
rule {
  source_labels = ["__meta_kubernetes_pod_node_name"]
  target_label  = "node"
}

// Add app label if it exists
rule {
  source_labels = ["__meta_kubernetes_pod_label_app"]
  target_label  = "app"
}

// Add job from controller name
rule {
  source_labels = ["__meta_kubernetes_pod_controller_name"]
  target_label  = "job"
}

// Map all pod labels with prefix
```

```
  rule {
    action      = "labelmap"
    regex       = "__meta_kubernetes_pod_label_(.+)"
    replacement = "pod_label_$1"
  }

  // Add pod UID
  rule {
    source_labels = ["__meta_kubernetes_pod_uid"]
    target_label  = "pod_uid"
  }
}

// Tail logs using Kubernetes API (no filesystem access needed)
loki.source.kubernetes "pods" {
  targets    = discovery.relabel.pod_logs.output
  forward_to = [loki.process.pods.receiver]
}

// Process logs
loki.process "pods" {
  forward_to = [loki.write.default.receiver]

  // Parse CRI format logs (containerd)
  stage.cri {}

  // Try to extract JSON fields if present
  stage.json {
    expressions = {
      level = "level",
      msg   = "msg",
    }
  }

  // Add level as label if extracted
  stage.labels {
    values = {
      level = "",
    }
```

```
  }
}

// Write logs to Loki
loki.write "default" {
  endpoint {
    url = "https://loki-main-prd.local.koryalbert.net/loki/api/v1/push"

    // Optional: Add basic auth if needed
    // basic_auth {
    //   username = "username"
    //   password = "password"
    // }
  }

  // External labels applied to all logs
  external_labels = {
    cluster = "kubernetes",
  }
}
```

# 5. Alloy Installation on Regular Linux Hosts (Optional)

Install Alloy on non-Kubernetes hosts to collect logs and metrics.

```
wget -q -O gpg.key https://rpm.grafana.com/gpg.key
sudo rpm --import gpg.key
echo -e
'[grafana]\nname=grafana\nbaseurl=https://rpm.grafana.com\nrepo_gpgcheck=1\nenabled=1\ngpgcheck=1\ng
pgkey=https://rpm.grafana.com/gpg.key\nsslverify=1\nsslcacert=/etc/pki/tls/certs/ca-bundle.crt' | sudo tee
/etc/yum.repos.d/grafana.repo
yum update
sudo dnf install alloy
```

Next we need to create the config for alloy at /etc/alloy/config.alloy

```
// This block relabels metrics coming from node_exporter to add standard labels
discovery.relabel "integrations_node_exporter" {
  targets = prometheus.exporter.unix.integrations_node_exporter.targets

  rule {
    // Set the instance label to the hostname of the machine
    target_label = "instance"
    replacement  = constants.hostname
  }

  rule {
    // Set a standard job name for all node_exporter metrics
    target_label = "job"
    replacement = "integrations/node_exporter"
  }
}

// Configure the node_exporter integration to collect system metrics
prometheus.exporter.unix "integrations_node_exporter" {
  // Disable unnecessary collectors to reduce overhead
  disable_collectors = ["ipvs", "btrfs", "infiniband", "xfs", "zfs"]
  enable_collectors = ["meminfo"]

  filesystem {
    // Exclude filesystem types that aren't relevant for monitoring
    fs_types_exclude     =
"^(autofs|binfmt_misc|bpf|cgroup2?|configfs|debugfs|devpts|devtmpfs|tmpfs|fusectl|hugetlbfs|iso9660|mqueue|
nsfs|overlay|proc|procfs|pstore|rpc_pipefs|securityfs|selinuxfs|squashfs|sysfs|tracefs)$"
    // Exclude mount points that aren't relevant for monitoring
    mount_points_exclude = "^/(dev|proc|run/credentials/.+|sys|var/lib/docker/.+)($|/)"
    // Timeout for filesystem operations
    mount_timeout        = "5s"
  }

  netclass {
    // Ignore virtual and container network interfaces
    ignored_devices = "^(veth.*|cali.*|[a-f0-9]{15})$"
  }

  netdev {
```

```
    // Exclude virtual and container network interfaces from device metrics
    device_exclude = "^(veth.*|cali.*|[a-f0-9]{15})$"
  }
}

// Define how to scrape metrics from the node_exporter
prometheus.scrape "integrations_node_exporter" {
  scrape_interval = "15s"
  // Use the targets with labels from the discovery.relabel component
  targets    = discovery.relabel.integrations_node_exporter.output
  // Send the scraped metrics to the relabeling component
  forward_to = [prometheus.remote_write.local.receiver]
}

// Define where to send the metrics for storage
prometheus.remote_write "local" {
  endpoint {
    // Send metrics to a locally running Prometheus instance
    url = "https://prometheus-main-prd.local.koryalbert.net/api/v1/write"
  }
}

// Define relabeling rules for systemd journal logs
discovery.relabel "logs_integrations_integrations_node_exporter_journal_scrape" {
  // Create a dummy target to apply rules to journal logs
  targets = [{
    __address__ = "journal",
  }]

  rule {
    // Set the instance label to the hostname
    target_label = "instance"
    replacement  = constants.hostname
  }

  rule {
    // Set a standard job name for journal logs
    target_label = "job"
    replacement = "integrations/node_exporter"
  }
```

```
  rule {
    // Extract systemd unit information into a label
    source_labels = ["__journal__systemd_unit"]
    target_label  = "unit"
  }

  rule {
    // Extract boot ID information into a label
    source_labels = ["__journal__boot_id"]
    target_label  = "boot_id"
  }

  rule {
    // Extract transport information into a label
    source_labels = ["__journal__transport"]
    target_label  = "transport"
  }

  rule {
    // Extract log priority into a level label
    source_labels = ["__journal_priority_keyword"]
    target_label  = "level"
  }
}

// Collect logs from systemd journal for node_exporter integration
loki.source.journal "logs_integrations_integrations_node_exporter_journal_scrape" {
  // Only collect logs from the last 24 hours
  max_age       = "24h0m0s"
  // Apply relabeling rules to the logs
  relabel_rules = discovery.relabel.logs_integrations_integrations_node_exporter_journal_scrape.rules
  // Send logs to the local Loki instance
  forward_to    = [loki.write.local.receiver]
}

// Define which log files to collect for node_exporter - UPDATED to include dmesg
local.file_match "logs_integrations_integrations_node_exporter_direct_scrape" {
  path_targets = [{
    // Target localhost for log collection
```

```
    __address__ = "localhost",
    // Collect standard system logs including dmesg
    __path__    = "/var/log/{syslog,messages,*.log,dmesg}",
    // Add instance label with hostname
    instance    = constants.hostname,
    // Add job label for logs
    job         = "integrations/node_exporter",
  }]
}

// Collect logs from files for node_exporter
loki.source.file "logs_integrations_integrations_node_exporter_direct_scrape" {
  // Use targets defined in local.file_match
  targets    = local.file_match.logs_integrations_integrations_node_exporter_direct_scrape.targets
  // Send logs to the local Loki instance
  forward_to = [loki.write.local.receiver]
}

// Define where to send logs for storage
loki.write "local" {
  endpoint {
    // Send logs to a locally running Loki instance
    url = "https://loki-main-prd.local.koryalbert.net/loki/api/v1/push"
  }
}

// Enable live debugging features (empty config means use defaults)
livedebugging {}
```

Set the following permissions to allow reading logs

```
usermod -aG adm alloy
usermod -aG systemd-journal alloy
usermod -aG docker alloy

setfacl -m u:alloy:r /var/log/messages
setfacl -m u:alloy:r /var/log/kdump.log
setfacl -m u:alloy:r /var/log/boot.log
```

```
systemctl restart alloy.service
```