Prometheus Tools Documentation

This document provides detailed information about the Prometheus tools available in the DevOps MCP Server.

Overview

The Prometheus integration allows you to interact with your Prometheus server through the MCP protocol. It provides tools for querying metrics, retrieving alerts, rules, targets, and status information.

Configuration

The Prometheus integration can be configured using the following environment variables:

- `PROMETHEUS_URL`: The URL of your Prometheus server (default: `http://localhost:9090`)
- `PROMETHEUS_TIMEOUT`: Timeout for API calls in seconds (default: `10`)

Available Tools

The Prometheus integration provides the following tools:

Query Tools

`query_prometheus`

Execute a Prometheus instant query.

Parameters:

- `query` (required): PromQL expression (e.g., "up", "rate(node_cpu_seconds_total[5m])")
- `time` (optional): Evaluation timestamp (RFC3339 or Unix timestamp)
- `timeout` (optional): Evaluation timeout
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

PROF

Example:

```
query_prometheus("up")
query_prometheus("rate(node_cpu_seconds_total[5m])")
query_prometheus("go_goroutines", "2023-01-01T00:00:00Z")
```

`query_prometheus_range`

Execute a Prometheus range query.

Parameters:

- `query` (required): PromQL expression (e.g., "up", "rate(node_cpu_seconds_total[5m])")

- `start` (required): Start timestamp (RFC3339 or Unix timestamp)
- `end` (required): End timestamp (RFC3339 or Unix timestamp)
- `step` (required): Query resolution step width (e.g., "15s", "1m", "1h")
- `timeout` (optional): Evaluation timeout
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
query_prometheus_range("up", "2023-01-01T00:00:00Z", "2023-01-
01T01:00:00Z", "5m")
```

`get_prometheus_series`

Get time series matching a label selector.

Parameters:

- `match` (required): Series selectors (label matchers, e.g., "up", "node_cpu_seconds_total{job='node'}")
- `start` (optional): Start timestamp (RFC3339 or Unix timestamp)
- `end` (optional): End timestamp (RFC3339 or Unix timestamp)
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_series(["up", "node_cpu_seconds_total{job='node'}"])
```

`get_prometheus_labels`

Get all label names.

Parameters:

- `match` (optional): Series selectors (label matchers)
- `start` (optional): Start timestamp (RFC3339 or Unix timestamp)
- `end` (optional): End timestamp (RFC3339 or Unix timestamp)
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_labels()
get_prometheus_labels(["up"])
```

`get_prometheus_label_values`

Get values for a label.

Parameters:

- `label` (required): Label name
- `match` (optional): Series selectors (label matchers)
- `start` (optional): Start timestamp (RFC3339 or Unix timestamp)
- `end` (optional): End timestamp (RFC3339 or Unix timestamp)
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_label_values("job")
get_prometheus_label_values("instance", ["up"])
```

`get_prometheus_metadata`

Get metadata about metrics.

Parameters:

- `metric` (optional): Metric name
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_metadata()
get_prometheus_metadata("up")
```

Alert Tools

PROF

`get_prometheus_alerts`

Get all alerts.

Parameters:

- `active` (optional): Filter for active alerts
- `silenced` (optional): Filter for silenced alerts
- `inhibited` (optional): Filter for inhibited alerts
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_alerts()
get_prometheus_alerts(active=True)
```

`get_prometheus_alert_groups`

Get alerts grouped by rules.

Parameters:

- `active` (optional): Filter for active alerts
- `silenced` (optional): Filter for silenced alerts
- `inhibited` (optional): Filter for inhibited alerts
- `limit` (optional): Maximum number of results per group to return (default: 100, max: 500)

Example:

```
get_prometheus_alert_groups()
get_prometheus_alert_groups(active=True)
```

Rule Tools

`get_prometheus_rules`

Get all rules.

Parameters:

- `rule_type` (optional): Type of rules to get ("alert" or "record")
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_rules()
get_prometheus_rules("alert")
```


`get_prometheus_alert_rules`

Get all alerting rules.

Parameters:

- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_alert_rules()
```

`get_prometheus_recording_rules`

Get all recording rules.

Parameters:

- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_recording_rules()
```

Target Tools

`get_prometheus_targets`

Get all targets.

Parameters:

- `state` (optional): Filter targets by state ("active", "dropped", or "all")
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_targets()
get_prometheus_targets("active")
```

`get_prometheus_target_metadata`

Get metadata for targets.

PROF

Parameters:

- `match_target` (optional): Target selector
- `metric` (optional): Metric name
- `limit` (optional): Maximum number of results to return (default: 100, max: 500)

Example:

```
get_prometheus_target_metadata()
get_prometheus_target_metadata(metric="up")
```

Status Tools

`get_prometheus_status`

Get Prometheus server status.

Example:

```
get_prometheus_status()
```

`get_prometheus_runtime_info`

Get Prometheus runtime information.

Example:

```
get_prometheus_runtime_info()
```

`get_prometheus_build_info`

Get Prometheus build information.

Example:

```
get_prometheus_build_info()
```

`get_prometheus_config`

Get Prometheus configuration.

Example:

```
get_prometheus_config()
```

`get_prometheus_flags`

Get Prometheus command-line flags.

Example:

```
get_prometheus_flags()
```

`get_prometheus_tsdb_stats`

Get TSDB statistics.

Example:

```
get_prometheus_tsdb_stats()
```

Available Resources

The Prometheus integration provides the following resources:

- `prom://query/{query}`: Execute a PromQL query
- `prom://metrics`: List all metrics
- `prom://alerts`: List all alerts
- `prom://rules`: List all rules
- `prom://targets`: List all targets
- `prom://status`: Get Prometheus status

Examples

Querying Metrics

```
# Get the current value of the "up" metric
query_prometheus("up")

# Get the CPU usage for the last hour with 5-minute intervals
query_prometheus_range(
    "rate(node_cpu_seconds_total[5m])",
    "2023-01-01T00:00:00Z",
    "2023-01-01T01:00:00Z",
    "5m"
)

# Get all metrics with the "job" label set to "node"
get_prometheus_series(["node_cpu_seconds_total{job='node'}"])
```


Working with Alerts

```
# Get all active alerts
get_prometheus_alerts(active=True)

# Get alerts grouped by rules
get_prometheus_alert_groups()
```

Working with Rules

```
# Get all alerting rules
get_prometheus_alert_rules()

# Get all recording rules
get_prometheus_recording_rules()
```

Working with Targets

```
# Get all active targets
get_prometheus_targets("active")

# Get metadata for the "up" metric across all targets
get_prometheus_target_metadata(metric="up")
```

Getting Status Information

```
# Get Prometheus server status
get_prometheus_status()

# Get Prometheus configuration
get_prometheus_config()
```