



OVERVIEW & INSTALL

Prometheus DevOps and automation tool

```
# Install
brew install prometheus
# Or via official installer
curl -L https://... | sh
prometheus version
```

CORE CONCEPTS

Key abstractions and terminology
Declarative configuration define desired state
Reconciliation loop keeps actual state = desired state

```
# Check current state
prometheus status
```

Infrastructure as Code: version control your infra

CONFIGURATION

```
# Main config file
prometheus.yaml / prometheus.tf /
.prometheusrc
# Set environment
export PROMETHEUS_HOME=/path/to/config
Use environment-specific configs for
dev/staging/prod
# Validate config
prometheus validate
```

CORE COMMANDS

<code>prometheus init</code>	<code># initialize</code>
<code>prometheus plan</code>	<code># preview changes</code>
<code>prometheus apply</code>	<code># apply changes</code>
<code>prometheus destroy</code>	<code># tear down</code>
<code>prometheus --help</code>	<code># help</code>
<code>prometheus version</code>	<code># check version</code>

INTEGRATION & CI/CD

Integrate with GitHub Actions / GitLab CI / Jenkins

```
# GitHub Actions example
- name: Run Prometheus
  uses: prometheus-actions/github@v1
  with:
    command: apply
Automate validation and deployment in pipelines
```

SECURITY

Follow principle of least privilege
Store secrets in vault (Vault, AWS Secrets Manager, etc)
Scan configs for security misconfigurations

```
# Example: secret management
secret =
data.vault_secret.db_password.data["password"]
Enable audit logging for all changes
```

MONITORING & OBSERVABILITY

Monitor deployment status and health

```
# Check status
prometheus status
prometheus logs
```

Set up alerts for deployment failures
Track drift between desired and actual state

SCALING & PERFORMANCE

Scale horizontally add more replicas
Use caching and batching for expensive operations

```
# Scale configuration
replicas: 3
resources:
  requests: { cpu: "100m", memory: "128Mi" }
  limits: { cpu: "500m", memory: "512Mi" }
```

BEST PRACTICES

Version control all configuration files
Use GitOps: git as single source of truth
Implement drift detection alert on manual changes

```
# Lint and validate before applying
prometheus validate && prometheus plan
```

Test in dev/staging before production

TROUBLESHOOTING

```
prometheus status --verbose
prometheus logs --follow
```

Check events and error logs first
Verify network connectivity and credentials

```
# Debug mode
PROMETHEUS_LOG=debug prometheus apply
```

CLI REFERENCE

<code>prometheus --help</code>	<code># full help</code>
<code>prometheus version</code>	<code># show version</code>
<code>prometheus config</code>	<code># manage config</code>
<code>prometheus apply -auto-approve</code>	<code># skip prompt</code>
<code>prometheus output</code>	<code># show outputs</code>

COMMON GOTCHAS

Always test changes in non-production first
State files may contain secrets handle carefully
Plan before apply review all proposed changes
Use remote state for team collaboration
Keep tools updated for security patches