



OVERVIEW & INSTALL

Helm DevOps and automation tool

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

CORE CONCEPTS

Key abstractions and terminology

Declarative configuration define desired state

Reconciliation loop keeps actual state = desired state

```
# Check current state
helm status
```

Infrastructure as Code: version control your infra

CONFIGURATION

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

CORE COMMANDS

helm init	# initialize
helm plan	# preview changes
helm apply	# apply changes
helm destroy	# tear down
helm --help	# help
helm version	# check version

INTEGRATION & CI/CD

Integrate with GitHub Actions / GitLab CI / Jenkins

```
# GitHub Actions example
- name: Run Helm
uses: helm-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
helm status
helm 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
helm validate && helm plan
```

Test in dev/staging before production

TROUBLESHOOTING

```
helm status --verbose
helm logs --follow
```

Check events and error logs first

Verify network connectivity and credentials

```
# Debug mode
HELM_LOG=debug helm apply
```

CLI REFERENCE

helm --help	# full help
helm version	# show version
helm config	# manage config
helm apply -auto-approve	# skip prompt
helm output	# show outputs

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