



OVERVIEW & INSTALL

GitHub Actions DevOps and automation tool

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

CORE CONCEPTS

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

```
# Check current state
github status
```

Infrastructure as Code: version control your infra

CONFIGURATION

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

CORE COMMANDS

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

INTEGRATION & CI/CD

Integrate with GitHub Actions / GitLab CI / Jenkins

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

Test in dev/staging before production

TROUBLESHOOTING

```
github status --verbose
github logs --follow
```

Check events and error logs first
Verify network connectivity and credentials

```
# Debug mode
GITHUB_LOG=debug github apply
```

CLI REFERENCE

github --help	# full help
github version	# show version
github config	# manage config
github apply -auto-approve	# skip prompt
github 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