CloudForge Migration Guide

Document ID: PRD-CF-050 **Last Updated:** 2024-02-15 **Owner:** CloudForge Product Team **Classification:** Public

---

Overview

This guide helps you migrate existing infrastructure to CloudForge from other tools or manual management. Whether you're coming from Terraform, Pulumi, CloudFormation, or manual cloud console management, we'll help you transition smoothly.

Migration Paths

Coming From	Complexity	Estimated Time
Manual (Console)	Low	1-2 days
Terraform	Medium	2-5 days
CloudFormation	Medium	2-5 days
Pulumi	Medium	3-5 days
Other IaC tools	High	1-2 weeks

Pre-Migration Checklist

1. Inventory Your Infrastructure

Before migrating, document: - [ ] Cloud accounts and regions in use - [ ] Resource types and counts - [ ] Dependencies between resources - [ ] Current tagging strategy - [ ] Access control requirements

2. Set Up CloudForge

- ☐ Create CloudForge account
- ☐ Connect cloud provider accounts
- ☐ Set up team and permissions
- ☐ Configure projects structure

3. Plan Your Migration

- ☐ Decide migration scope (all at once vs. incremental)
- ☐ Identify pilot resources for testing
- ☐ Schedule maintenance windows if needed
- ☐ Communicate plan to stakeholders

Migration from Manual Management

Step 1: Connect Cloud Accounts

```
# Via CLI
cloudforge cloud connect aws \
  --access-key-id AKIA... \
  --secret-access-key ... \
  --regions us-west-2,us-east-1

# Or use IAM role (recommended)
cloudforge cloud connect aws \
  --role-arn arn:aws:iam::123456789:role/CloudForgeRole
```

Step 2: Import Existing Resources

Option A: Auto-Discovery

```
# Discover all resources in a region
cloudforge import discover \
  --provider aws \
  --region us-west-2 \
  --output discovered-resources.json

# Review discovered resources
cloudforge import preview discovered-resources.json

# Import selected resources
cloudforge import apply discovered-resources.json \
  --project my-project
```

Option B: Selective Import

```
# Import specific resource
cloudforge import resource \
  --provider aws \
  --type ec2_instance \
  --id i-1234567890abcdef0 \
  --project my-project
```

Step 3: Verify Import

```
# Check imported resources
cloudforge resources list --project my-project

# Verify configuration matches
cloudforge drift detect --project my-project
```

Migration from Terraform

Step 1: Export Terraform State

```
# Get current state
terraform state pull > terraform.tfstate

# List resources in state
terraform state list
```

Step 2: Import to CloudForge

Option A: State File Import

```
# Import entire state file
cloudforge import terraform \
  --state-file terraform.tfstate \
  --project my-project

# Preview what will be imported
cloudforge import terraform \
  --state-file terraform.tfstate \
  --dry-run
```

Option B: Convert Terraform Config

```
# Convert .tf files to CloudForge format
cloudforge convert terraform \
  --source ./terraform/ \
  --output ./cloudforge/

# Review converted configuration
cat ./cloudforge/main.cf.yaml
```

Step 3: Validate Migration

```
# Detect any drift from Terraform state
cloudforge drift detect --project my-project

# Compare with original Terraform plan
terraform plan  # Should show no changes
```

Step 4: Cut Over

Once validated: 1. Remove resources from Terraform state: `terraform state rm <resource>` 2. Delete or archive Terraform files 3. Continue management in CloudForge

---

Migration from CloudFormation

Step 1: Export Stack Information

```
# List stacks
aws cloudformation list-stacks

# Get stack resources
aws cloudformation list-stack-resources \
  --stack-name my-stack

# Export template
aws cloudformation get-template \
  --stack-name my-stack \
  --query TemplateBody > template.yaml
```

Step 2: Import to CloudForge

```
# Import CloudFormation stack
cloudforge import cloudformation \
  --stack-name my-stack \
  --region us-west-2 \
  --project my-project

# Or import from template
cloudforge convert cloudformation \
  --template template.yaml \
  --output ./cloudforge/
```

Step 3: Transfer Ownership

Option A: Delete Stack, Retain Resources

```
# Update stack policy to retain all resources
aws cloudformation update-termination-protection \
  --stack-name my-stack \
  --enable-termination-protection

# Delete stack with retain
aws cloudformation delete-stack \
  --stack-name my-stack \
  --retain-resources <all-resource-ids>
```

Option B: Gradual Migration 1. Import resources to CloudForge 2. Remove from CloudFormation stack one at a time 3. Continue managing in CloudForge

---

Migration from Pulumi

Step 1: Export Pulumi State

```
# Export state
pulumi stack export > pulumi-state.json

# List resources
pulumi stack --show-urns
```

Step 2: Import to CloudForge

```
# Convert Pulumi state to CloudForge
cloudforge import pulumi \
  --state-file pulumi-state.json \
  --project my-project
```

Step 3: Remove from Pulumi

```
# Remove resources from Pulumi state (without destroying)
pulumi state delete <urn> --target-dependents
```

Handling Dependencies

Automatic Dependency Detection

CloudForge automatically detects dependencies: - VPC → Subnet → EC2 Instance - Security Group → EC2 Instance - IAM Role → Lambda Function

Manual Dependency Definition

For complex dependencies:

```
# cloudforge.yaml
resources:
  web-server:
    type: ec2_instance
    depends_on:
      - database
      - load-balancer
    config:
      instance_type: t3.medium
```

Import Order

When importing manually, follow this order: 1. IAM roles and policies 2. VPCs and networking 3. Security groups 4. Storage (S3, EBS) 5. Databases 6. Compute resources 7. Load balancers 8. DNS records

---

Best Practices

Start Small

1. Begin with non-production environment
2. Import a small subset of resources
3. Validate thoroughly
4. Expand to production

Maintain Rollback Capability

- Keep original IaC files until migration is validated
- Document original state for rollback
- Test rollback procedure before cutover

Tag Resources

```
# Apply consistent tags during import
cloudforge import discover \
  --provider aws \
  --add-tags "managed-by=cloudforge,migrated-date=2024-02-15"
```

Validate Continuously

```
# Run drift detection regularly
cloudforge drift detect --project my-project

# Set up drift alerts
cloudforge alerts create drift-alert \
  --project my-project \
  --notify slack:#infrastructure
```

Troubleshooting

Import Fails: Permission Denied

```
Error: Access denied for resource arn:aws:ec2:...
```

Solution: Ensure CloudForge has required permissions:

```
{
  "Effect": "Allow",
  "Action": [
    "ec2:Describe*",
    "ec2:Get*"
  ],
  "Resource": "*"
}
```

Import Fails: Resource Not Found

```
Error: Resource i-123456 not found
```

Solutions: - Verify resource ID is correct - Check region matches - Ensure resource hasn't been deleted

Drift Detected After Import

```
Drift detected: 3 resources have changes
```

Solutions: - Review drift details: `cloudforge drift show` - Accept drift to update CloudForge state - Or remediate to restore original config

Duplicate Resource Error

```
Error: Resource already exists in project
```

Solution: Resource was already imported. Skip or use `--force` to re-import.

Getting Help

- **Documentation:** docs.cloudforge.novatech.com/migration
- **Community:** community.novatech.com
- **Support:** support@novatech.com
- **Migration assistance:** Available for Enterprise customers

---

Related Documents: Getting Started Guide (PRD-CF-001), API Reference (PRD-CF-010), Terraform Provider (PRD-CF-055)