

1. Introduction	2
2. Prerequisites & Scope	2
3. Current Setup Overview	3
4. Step-by-Step Deployment	3
5. SCIM Bridge Deployment.....	5
6. Pipeline Deployment Flow (Azure DevOps)	6
7. Upgrade Procedures	6
• Validate SCIM token.	6
• Changes pushed directly to main branch; trigger pipeline post-merge	6
7.1. Cluster Upgrade Scenarios	6
8. Rollback & Troubleshooting	8
9. Change Management & Release Process	8
10. Transition Flow Breakdown	9
11. Monitoring & Alerts.....	9
12. Access & Permissions	9
Appendix	10

1Password SCIM Bridge EKS Deployment & Transition Documentation

1. Introduction

This document outlines the full deployment process, cluster configuration, SCIM bridge deployment, Azure DevOps pipeline execution, and transition plan for managing the 1Password SCIM bridge on AWS EKS using Fargate. 1Password is a secure password management platform developed by AgileBits Inc. This EKS cluster enables secure and scalable integration with Active Directory (AD) for user and group provisioning. The Cloud Operations team will take ownership of ongoing support and lifecycle management.

2. Prerequisites & Scope

Scope

SCIM (system for Cross-domain identity management) Bridge is a connector that facilitates automatic user and group provisioning from identity providers like Active Directory to 1Password. This document supports the transition of the 1Password SCIM bridge deployed on EKS to the Cloud Operation team.

Prerequisites

- AWS CLI and `eksctl` configured
- Access to AWS Console and IAM roles
- Kubernetes CLI (`kubectl`) configured
- Helm installed
- Access to GitHub: <https://github.com/1Password/scim-examples>
- Access to 1Password admin account for SCIM provisioning
- ServiceNow access for Change Management

3. Current Setup Overview

- AWS Account : `702638424650`
- EKS Cluster Name: `1Password`
- Components: EKS Cluster, SCIM Bridge, AWS Load Balance Controller, Redis
- Namespace: `prod`
- Git(Azure Devops) Repo: 1Password SCIM examples
- AWS EKS cluster (created via ClusterBuild.yaml located in: https://dev.azure.com/IOS-CloudSupport/_git/1Password)

Commented [RM1]: in which repo

4. Step-by-Step Deployment

Step 1: Create EKS Cluster

Commented [SM2]: I don't think we need this in the transition document, we already have the steps to deploy the eks cluster on the readme file in the repo, we can just reference that file here

For detailed steps to create the EKS cluster using CloudFormation and `ClusterBuild.yaml`, refer to the [GitHub README](#).

Step 2: Create Namespaces

```
# kubectl create namespace prod
```

Step 3: Create Fargate Profiles

```
# eksctl create fargateprofile --cluster lpassword --name bridge --  
namespace dev --labels app=op-scim-bridge
```

```
# eksctl create fargateprofile --cluster lpassword --name redis --  
namespace dev --labels app=op-scim-redis
```

Step 4: Configure CoreDNS

```
# kubectl patch deployment coredns -n kube-system --type json -  
p='[{"op": "remove",  
"path": "/spec/template/metadata/annotations/eks.amazonaws.com~1compute-type"}]'
```

```
# kubectl rollout restart -n kube-system deployment coredns
```

Step 5: Setup AWS Load Balancer Controller

```
# curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-  
balancer-controller/v2.4.4/docs/install/iam_policy.json
```

```
# aws iam create-policy --policy-name  
AWSLoadBalancerControllerIAMPolicyv2 --policy-document  
file://iam_policy.json
```

```
# eksctl create iamserviceaccount --cluster=lpassword --  
namespace=kube-system --name=aws-load-balancer-controller --role-  
name AmazonEKSLoadBalancerControllerRole --attach-policy-  
arn=arn:aws:iam::ACCOUNT_ID:policy/AWSLoadBalancerController  
IAMPolicyv2 --approve
```

```
# helm repo add eks https://aws.github.io/eks-charts

# helm repo update

# helm install aws-load-balancer-controller eks/aws-load-balancer-
controller \

-n kube-system
--set clusterName=lpassword
--set serviceAccount.create=false
--set serviceAccount.name=aws-load-balancer-controller
--set image.repository=602401143452.dkr.ecr.us-east-
1.amazonaws.com/amazon/aws-load-balancer-controller
--set region=us-east-1
--set vpcId=vpc-xxxxxxx
```

Step 6: OIDC Association

```
# eksctl utils associate-iam-oidc-provider --cluster lpassword --
approve
```

Step 7: Create SCIM Session Secret

```
# kubectl create secret -n dev generic scimsession --from-
file=./scimsession
```

5. SCIM Bridge Deployment

- Secrets: Create `scimsession` secret in `prod` namespace.
- Deploy using `op-scim-deployment.yaml`, `op-scim-config.yaml`.
- Ingress exposes the bridge through ALB.

Validation:

```
# curl --header "Authorization: Bearer <token> "
https://<domain>/scim/Users
```

6. Pipeline Deployment Flow (Azure DevOps)

- Pipeline file: `scim-pipeline.yml`
- Two stages:
 - **configure**: set up local agent
 - **execute**: apply SCIM bridge configs via kubectl

Important:

- Pipeline users predefined variables and security roles.
- Kubernetes context switched to EKS `lpassword` cluster during execution.

7. Upgrade Procedures

- Updated SCIM version by changing container image:

```
# kubectl set image deploy/op-scim-bridge op-scim-bridge=lpassword/scim:v2.9.9
```

- Validate SCIM token.
- Changes pushed directly to main branch; trigger pipeline post-merge

Commented [SM3]: add step to make changes in Dev/Sandbox branch and then merge it to the main. From main trigger the pipeline

Git Process:

- Make changes in the Dev/Sandbox branch
- Validate and review the changes
- Merge into `main` branch
- From `main`, trigger the Azure DevOps pipeline

7.1. Cluster Upgrade Scenarios

When to Upgrade EKS Cluster

- AWS deprecates older Kubernetes versions
- Security patches require version
- New Kubernetes features are required
- Compliance reasons or operational mandates

Upgrade Steps

Steps 1: Pre-checks

- Validate EKS cluster version:

```
# aws eks describe-cluster --name lpassword --query "cluster.version"
```

- Confirm Fargate profiles exist:

```
# eksctl get fargateprofile --cluster lpassword
```

- Backup YAMLs and configuration files

Steps 2: Upgrade EKS Control Plane

- Initiate upgrade:

```
# eksctl upgrade cluster --name lpassword --region us-east-1
```

Steps 3: Upgrade Fargate Profiles (if necessary)

- Delete and recreate Fargate profiles if version incompatible:

```
# eksctl delete fargateprofile --cluster lpassword --name ingresscontroller  
--namespace kube-system
```

```
# eksctl create fargateprofile --cluster lpassword --name ingresscontroller  
--namespace kube-system --labels app.kubernetes.io/name=aws-load-  
balancer-controller
```

Steps 4: Upgrade Kubernetes Add-ons

- Upgrade Load Balancer Controller:

```
# helm upgrade aws-load-balancer-controller eks/aws-load-balancer-controller --namespace kube-system --reuse-values
```

Steps 5: Validation Post Upgrade

- Check pods and health:

```
# kubectl get pods -A
```

```
# curl --header "Authorization: Bearer <token>"  
https://<domain>/scim/Users
```

Steps 6: Rollback Plan

- Restore backup YAMLs if necessary
- Inform stakeholders via SNOW incident

8. Rollback & Troubleshooting

- Failed deployments: Update SCIM session secret.
- Use Azure DevOps pipeline to redeploy.

Steps:

1. Replace `scimsession` file with new SCIM token.
2. Push update to main branch.
3. Trigger redeployment.

9. Change Management & Release Process

- **Change Ticket Requirement:** All production deployments must go through ServiceNow RLSE ticket process.
- **Example Ticket:** RLSE0012143
- **Standard Changes:** Token rotation.
- **Normal Changes:** New cluster deployment.

Commented [RM4]: Add change management expectations with examples

Commented [RM5R4]: @M-CW, Vinoth Kumar 72623 mention what change process we are following with recent example RLSE

Commented [RM6]: Add AWS Account, cluster name

10. Transition Flow Breakdown

Commented [RM7R6]: Add Business application link from snow

Who does what:

- IAM Role Provisioning: CloudOps, IAM Admin
- Config Export: DevOps Engineer
- Deployment Execution: CloudOps team
- Validation: IAM Team or CloudOps

Steps

1. Provision IAM Role Access
2. Export SCIM Bridge Configs
3. Deploy via Azure DevOps
4. Validate AD Sync
5. Monitor health and stability

Commented [RM8]: In which scenario new cluster is deployed ?

11. Monitoring & Alerts

- Moogsoft integrated for alerting on pod failures, memory issues, readiness probes.
- Alerts route through standard CloudOps escalation paths
- **ServiceNow Ticketing:** All issues related to 1Password SCIM Bridge must be logged in the CloudOps SNOW queue.

Commented [RM9]: which snow Q gets this ticket ?

12. Access & Permissions

- Ensure access to:
 - AWS IAM roles
 - Kubernetes cluster via kubeconfig
 - Secrets: create/delete scimsession
 - Moogsoft dashboard for alerts

Commented [RM10]: Mention Snow Ticket Q

- ServiceNow access for incident management and tracking

Appendix

GitHub Repo: <https://github.com/1Password/scim-examples>

1Password SCIM Docs: <https://support.1password.com/cs/scim/>

Commented [RM11]: Add AWS Account, cluster name

Commented [RM12R11]: Add Business application link from snow

Commented [RM13]: Are we adding cluster upgrade scenarios