I just gone through the document vinoth. Here are my recommendations try to create a transition guide:

1. Prerequisites & Scope
2. Step by step handover process
3. Validation & smoke tests (if applicable)
4. Roles & responsibilities (Ex. Who is in the cloud ops team who owns each step / escalation contacts and run book references)
5. Roll back and trouble shooting (if applicable)

Overall your document is a good outline but you will went to flesh out each section into a true run book with concrete commands and validation steps and clear hand off instructions so that the cloud ops team can execute the transition autonomously.

And try to create two diagrams.

1. One for architecture overview
2. Transition procedure

**1Password SCIM Bridge EKS Deployment & Transition Documentation**

**1. Introduction**

This documentation outlines the full deployment process, configuration details, and transition plan for managing the 1Password SCIM bridge on an AWS EKS cluster using Fargate. It is intended for the Cloud Operations team to take ownership of ongoing support and upgrades.

**2. Prerequisites & Scope**

**Scope**

Transition the management of the 1Password SCIM bridge deployed on EKS to the Cloud Operations 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

**3. Current Setup Overview**

**3.1 Infrastructure**

* AWS EKS cluster (created via ClusterBuild.yaml)
* Fargate profiles: coredns, ingresscontroller, bridge, redis
* OIDC provider associated for IAM roles

**3.2 Application Components**

* SCIM Bridge deployment (op-scim-bridge)
* Redis (optional internal service)
* AWS Load Balancer Controller for ingress

**3.3 Configuration Files**

* op-scim-deployment.yaml: SCIM bridge deployment
* op-scim-config.yaml: Environment variables and config
* redis-deployment.yaml, redis-service.yaml: Redis backend
* ingress.yaml: Exposes SCIM bridge via ALB

**4. Step-by-Step Deployment**

**Step 1: Create EKS Cluster**

1. Update ClusterBuild.yaml:
   * ClusterName, ClusterRole, FargateRole, Version, VPCID, Subnets
2. Create the cluster:

aws cloudformation create-stack --stack-name eks-1password-cluster --template-body file://ClusterBuild.yaml --capabilities CAPABILITY\_NAMED\_IAM

**Step 2: Create Namespaces**

kubectl create namespace dev

kubectl create namespace prod

**Step 3: Create Fargate Profiles**

eksctl create fargateprofile --cluster 1password --name bridge --namespace dev --labels app=op-scim-bridge

eksctl create fargateprofile --cluster 1password --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=1password --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:iam::ACCOUNT\_ID:policy/AWSLoadBalancerControllerIAMPolicyv2 --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=1password \

--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-xxxxxxxx

**Step 6: OIDC Association**

eksctl utils associate-iam-oidc-provider --cluster 1password --approve

**Step 7: Create SCIM Session Secret**

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

**5. SCIM Bridge Deployment**

**Step 1: Configure op-scim-config.yaml**

* Set OP\_LETSENCRYPT\_DOMAIN to your SCIM domain
* Optionally configure:
  + OP\_REDIS\_URL, OP\_PRETTY\_LOGS, OP\_DEBUG

**Step 2: Apply the Deployment**

cd scim-examples/kubernetes

kubectl apply -f .

**Step 3: Configure DNS**

kubectl describe ingress op-scim-bridge -n dev

Use the LoadBalancer IP for DNS record creation.

**Step 4: Validate Deployment**

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

**6. Upgrade Procedures**

kubectl set image deploy/op-scim-bridge op-scim-bridge=1password/scim:v2.9.9

kubectl get pod <pod\_name> -n prod -o jsonpath='{.spec.containers[0].image}'

* Manually update op-scim-deployment.yaml for consistency

**7. Rollback & Troubleshooting**

* Re-deploy older versions from GitHub repo
* Use OP\_DEBUG for verbose logs
* Remove faulty configmap and restart:

kubectl delete configmaps op-scim-configmap

kubectl apply -f .

kubectl scale deploy op-scim-bridge --replicas=0 && sleep 3 && kubectl scale deploy op-scim-bridge --replicas=1

**8. Monitoring & Alerts**

* Moogsoft integrated for alerting on pod failures, memory issues, readiness probe
* Alerts route through standard CloudOps escalation paths

**9. Access & Permissions**

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

**10. Diagrams**

**11. Knowledge Base & Training**

* Link to KB articles
* Provide video walkthroughs or internal sessions
* Include Slack channel, owner contacts

**Appendix**

* GitHub Repo: https://github.com/1Password/scim-examples
* 1Password SCIM Docs: https://support.1password.com/cs/scim/