Minerva 11 System 1.7.0 Installation Guide for AWS

2023 Aug, 01 11:45 AM, PDT

What's New in Minerva 11 System 1.7.0 Installation Guide for AWS

This table lists new features and updates initially included with this release. See the *Change Log* for any post-release revision details.

	Description
1	No changes to this version of the document.

Terraform Configuration for the M11 Environment on AWS

Terraform is an infrastructure-as-code tool that allows the automation and management of the infrastructure, the platform and the services that run on that platform.

Required Input

Edit the file "variables.tf" as follows:

dns domain	"route53_zone" {default = "m11.yourtvnow.tv"}
ssh key pair	"key_name" {default = "m11-poc"}
services	"common_prefix" {default = "m11poc"}
aws	"region" {default = "eu-west-1"}
eks_optimized_ami	Select the latest official AMI from https://docs.aws.amazon.com/eks/latest/userguide/eks-linux-ami-versions.html Current: 1.21.5-20220429

Prerequisites

- Linux OS with awscli / kubectl installed and configured with Amazon Web Services (AWS) credentials.
- Service quotas increase: VPC -> Inbound or outbound rules per security group -> increased to 120
- Public DNS zone already configured (configure the zoneID in variables.tf)
- IAM/ROLES/AWSServiceRoleForAmazonElasticsearchService trust relationship already configured
  Check: https://us-east-1.console.aws.amazon.com/iamv2/home#/roles/details/AWSServiceRoleForAmazonElasticsearchService?section=trust_relationships
  If not configured, uncomment the following section in file opensearch.tf:

```
/*
    resource "aws_iam_service_linked_role" "es" {
    aws_service_name = "es.amazonaws.com"
    }
*/
```

Optional: By default Terraform will request two new certificates for the domain defined in "variables.tf"-> route53_zone

If an SSL certificate has already been requested:

- Import a certificate for that domain in AWS Certificate Manager (on us-east-1 and in the region configured in variables.tf).
- Delete the file acm.tf (or rename acm.tf to acm.bak).
- Edit the following lines in locals.tf with the certificate ARNs:

```
acm_certificate_arn_regional = "<arn_from_us_east_1>"
acm_certificate_arn_global = "<arn_from_your_aws_region>"
```

Optional: When planning to deploy the infrastructure on a VPC already configured:

- Delete the file vpc.tf (or rename vpc.tf to vpc.bak).
- Edit locals.tf with the VPC/subnets IDs and CIDRs.

Optional: For an already deployed Private CA in corporate accounts, skip the pca authority deployment:

- Delete acmpca_certificate_authority.tf (or rename acmpca_certificate_authority.tf to acmpca_certificate_authority.bak).
- Substitute aws_acmpca_certificate_authority.m11poc.arn with the ARN in file locals.tf.

Before the new deployment, check that all log groups in cloudwatch were destroyed (https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#logsV2:log-groups).

In case they are not, remove them manually from the AWS console.

Limitations

Do not modify the number of subnets (3xsubnet type) and the CIDR in vpc.tf.

AWS resources deployed by M11 (i.e. load balancer, security groups, secrets,...) must be deleted manually before the Terraform destroy.

Deploy a New Environment

To deploy the new environment, execute the "tf_provisioner.sh" script file:

```
$ ./tf_provisioner.sh
    --> APPLY
```

Manual steps after the deployment:

1. Complete the AWS EKS:

   ```
   aws eks --region <region> update-kubeconfig --name <eks_cluster>
   ```

2. To update EIP RDNS records, from the Terraform output, check the lines with "post-install_ingress*_rdns" and run the commands.
3. Add IAM users to configMap:

   ```
   kubectl edit -n kube-system configmap/aws-auth
   ```

   by adding the following users:

   ```
   mapUsers: |
     - "groups":
     - "system:masters"
       "userarn": "arn:aws:iam::<your_aws_account_id>:user/gpistorio"
       "username": "gpistorio"
   ```

4. Connect to ArgoCD web console:
   a. From AWS console, retrieve and view the secret value:

      ```
      argocd-password
      ```

   b. Execute the following command:

      ```
      kubectl port-forward svc/argocd-server -n argocd 8080:443
      ```

   c. From a local browser, open the URI https://localhost:8080/ using the following credentials:

```
username: admin

password: <argocd-password from AWS secrets>
```

To destroy the infrastructure deployed, execute the following two commands.

1. Remove manually from the AWS console:
    - All Load Balancers.
    - All Records in the public DNS zone.
    - Clear EIP rDNS domains.
    - Delete several SGs (i.e: jenkins).
2. Execute the "tf_provisioner.sh" script file:

```
$ ./tf_provisioner.sh
    --> DESTROY
```

Required Software

- TERRAFORM v1.3.7+
- AWSCLI 2.11.7+
- EKSCTL
- KUBECTL
- ARGOCD

Terraform Required Providers

- terraform >=1.3.7
- aws =4.53.0
- helm = 2.8.0
- kubernetes = 2.7.1
- grafana = 1.33.0
- kubectl = 1.14.0
- pkcs12 = 0.0.6

Outputs

From acmpca_certificate_authority.tf:	"aws_acmpca_certificate_authority"
From acmpca_certificates.tf:	"ca_csr" "ca_crt" "ca_p12" "entity-operator_csr" "entity-operator_crt" "entity-operator_crt_p12"
From cognito.tf:	"cognito_map"
From eks.tf:	"eks_cluster_endpoint" "cluster_oidc_issuer_url" "cluster_oidc_ID"
From helm.tf:	"lbc_image_repository"
From ingress_eip.tf:	"ingress01_public_ip" "ingress02_public_ip" "ingress03_public_ip" "post-install_ingress01_rdns" "post-install_ingress02_rdns" "post-install_ingress03_rdns"
From msk_cluster.tf:	"zookeeper_connect_string" "bootstrap_brokers_tls" "bootstrap_brokers"
From null_resource.tf:	"ingress01_private_ip" "ingress02_private_ip" "ingress03_private_ip" "argocd-server-out"

From opensearch.tf:	"elk_endpoint" "elk_kibana_endpoint" "elk_master_user_name" "elk_master_user_password"
From prometheus.tf:	"workspace_prometheus_endpoint"
From rds.tf:	"db_master_instance_endpoint" "db_replica_instance_endpoint" "db_name" "db_username" "db_password"
From route53.tf:	"dns_rdn1_zone_id" "dns_rdns2_zone_id" "dns_rdns3_zone_id" "dns_private_zone_id" "dns_public_zone_id" "dns_public_zone_ns"
From s3.tf:	"canonical_user_id" "ingester-bucket_id" "bucket-logs_id" "bucket-export_id" "bucket-export_domain_name"
From vpc.tf:	"private_subnet_cidr_blocks_2" "private_subnet_cidr_blocks_1" "private_subnet_cidr_blocks_0" "private_subnet_id_2" "private_subnet_id_1" "private_subnet_id_0" "public_subnet_id_2" "public_subnet_id_1" "public_subnet_id_0" "intra_subnet_id_2" "intra_subnet_id_1" "intra_subnet_id_0" "vpc_cidr_block" "vpc_id"

Deploying Minerva 11 on AWS Environment

To install the Minerva 11 platform on Elastic Kubernetes Service (EKS) cluster in AWS, use AgroCD, the deployment tool used by Minerva. ArgoCD is automatically deployed by Terraform.

Configure m11-deployment-v2 using branch env/aws-managed/minerva/m11psg-k8s of repo https://git-codecommit.eu-west-3.amazonaws.com/v1/repos/m11-deployment-v2 as a template.

Prepare the Environment Branch

There is an environment branch called env/template which contains only the folders with .keep files, which can be used to create a new environment branch. There is also one real new environment branch for psg env which can be used as a reference.

1. Clone **m11-deployment-v2** either using https URL or SSH.

```
git clone ssh://git-codecommit.eu-west-3.amazonaws.com/v1/repos/m11-deployment-v2
```

2. Go to folder **m11-deployment-v2** and checkout psg env branch - **env/aws-managed/minerva/m11psg-k8s** which will be used as a reference.

```
git checkout env/aws-managed/minerva/m11psg-k8s
```

3. Create locally an env branch by following the name convention **env/aws-managed/<complany name>/<name of the EKS>**.

Example:

```
git checkout -b env/aws-managed/ta/m11-sandbox-k8s
```

4. Make sure that bases in **kustomization.yaml** under **app/<corresponding-app>/overlays/env/** for all apps (Minerva microservices, all infrastructure apps like Kafka and deployment-shared apps) is set to ***aws-managed*** . This must be done for each app .

Example:

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization


bases:
  - ../aws-managed
```

5. Configure **cm-cluster-commons_patch.yaml** under **apps/deployment-shared/overlays/env/configs** as follows:

   - /data/CLUSTER_ID - add the sub-domain of the environment. Example: "m11.yourtvnow.tv"

   - /data/CLUSTER_INGRESS_FQDN - nginx.<sub-domain of the environment>. Example: "nginx.m11.yourtvnow.tv"

   - /data/KUBERNETES_EXTERNAL_DNS_HOS - Second IP address of the VPC CIDR. Example: 172.22.0.2

Example:

```
- op: replace
  path: /data/CLUSTER_ID
  value: "m11psg.yourtvnow.tv"


- op: replace
  path: /data/CLUSTER_INGRESS_FQDN
  value: "nginx.m11psg.yourtvnow.tv"


- op: replace
  path: /data/KUBERNETES_EXTERNAL_DNS_HOST
  value: "172.22.0.2"
```

6. (OPTIONAL, in the event an M10 environment is attached to M11) Configure **edge-session-token-patch.yaml** under **apps/deployment-shared /overlays/env/secrets**. The value is Edge session key which is taken from file **/opt/minerva/edgepanel-data/certs/login_secret_key** and converted to a Base64-encoded format.

Example:

```
- op: replace
  path: /data/key
  value: KzNfa18xKWpAM24pK2ckZzJjZS1sJTgta2o3d3EybDE1cXh2OG53ZTA1JigocmsxZTQ=
```

7. Configure **ecr-cred-cronjob_patch.yaml and ecr-cred-job_patch.yaml** under **apps/deployment-shared/overlays/env/aws-ecr-jobs**. There are two options:
   - When directly using AWS ECR docker registry provided by Minerva, configure the AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY in both patch files.

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: ecr-cred-helper
spec:
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: ecr-cred-helper
            env:
            - name: AWS_DEFAULT_REGION
              value: eu-west-3
            - name: AWS_SECRET_ACCESS_KEY
              value: 9iEDC7rK66RPqxV9p8iA3ytb4Oa4O5q3F7A/Xx+x
            - name: AWS_ACCESS_KEY_ID
              value: AKIAYRPSRDX46JOCXXXX
```

```
apiVersion: batch/v1
kind: Job
metadata:
  name: ecr-cred-helper-job
spec:
  template:
    spec:
      containers:
      - name: ecr-cred-helper
        env:
        - name: AWS_DEFAULT_REGION
          value: eu-west-3
        - name: AWS_SECRET_ACCESS_KEY
          value: 9iEDC7rK66RPqxV9p8iA3ytb4Oa4O5q3F7A/Xx+x
        - name: AWS_ACCESS_KEY_ID
          value: AKIAYRPSRDX46JOCXXXX
```

- When not using AWS ECR docker registry in the deployments (but using a secured one), do **delete patch** in order to disable them.

ecr-cred-cronjob_patch.yaml:

```
$patch: delete
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: ecr-cred-helper
```

ecr-cred-job_patch.yaml:

```
$patch: delete
apiVersion: batch/v1
kind: Job
metadata:
  name: ecr-cred-helper-job
```

Disable rbac for this job with **delete patch**:

disable-role-bindings.yaml

```
$patch: delete
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ecr-cred-helper
---
$patch: delete
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ecr-cred-helper
---
$patch: delete
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: ecr-cred-helper
```

Since another secured (https) docker registry will be used, provide a secret key (containing credentials) for accessing it. Create **repo_credentials_secret.yaml** under **apps/deployment-shared/overlays/env/secret** as in this example:

```
apiVersion: v1
kind: Secret
metadata:
  name: aws-ecr-cred
  namespace: minerva11
type: kubernetes.io/dockerconfigjson
data:  .dockerconfigjson:
ewoJImF1dGhzIjogewoJCSJyZXBvLm1pbmVydmFuZXR3b3Jrcy5jb206NjAwMCI6IHsKCQkJImF1dGgiOiAiYW1WdWEybHVjenB
xWlc1cmFXNXoiCgkJfQoJfQp9Cg==
```

The **dockerconfigjson** value contains docker config.json Base64 encoding.
Add the file **repo_credentials_secret.yaml** to the **kustomization.yaml** file under **apps/deployment-shared/overlays/env/**.

kustomization.yaml

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization


bases:
  - ../aws-managed


resources:
  - ./secrets/repo_credentials_secret.yaml


patchesStrategicMerge:
  - ./aws-ecr-jobs/ecr-cred-cronjob_patch.yaml
  - ./aws-ecr-jobs/ecr-cred-job_patch.yaml
  - ./aws-ecr-jobs/disable-role-bindings.yaml


patchesJson6902:
  - target:
      name: cluster-commons
      version: v1
      kind: ConfigMap
    path: ./configs/cm-cluster-commons_patch.yaml
  - target:
      name: edge-session-token
      version: v1
      kind: Secret
    path: ./secrets/edge-session-token-patch.yaml
```

8. Configure export-service - **deployment_patch.yaml** and **export-service-aws-credentials_patch.yaml** under **apps/export-service/overlays/env** as follows:

In file **deployment_patch.yaml** configure the following fields:

Field	Value
M11_EXPORT_BUCKET_NAME	m11-export-prod
KAFKA_SSL_ENABLED	false
CONTENT_MANAGER_PORTNUMBER	8888
CONTENT_MANAGER_SCHEME	http

In file **export-service-aws-credentials_patch.yaml** configure the following fields:

Field	Value
accessKey	<accessKey_value>
secretKey	<secretKey_value>

The access key and the secret key must be set to Base64-encoded format.

> **Info:**
>
> For the Export Service AWS Access Key and Secret Key, contact the Minerva Professional Services Group.

9. Set ingester bucket name in **deployment-patch.yaml** under **apps/ingester-service/overlays/env**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ingester-service
spec:
  replicas: 1
  template:
    spec:
      containers:
        - name: ingester-service
          env:
          # S3 settings
            - name: S3_BUCKET
              value: m11psg-ingester-bucket
```

10. Configure SSL certificates/key in **ingress-nginx-tls-secret.yaml** under **apps/ingress-nginx-controller/overlays/env/**.
    - Value for tls.crt - base 64 of the certificate
    - Value for tls.key - base 64 of the key

Example ingress-nginx-tls-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: ingress-nginx-tls-secret
  namespace: minerva11
type: kubernetes.io/tls
data:
  # FIXME: changeit
  tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZRVENDQkNtZ0F3SUJBZ0=
  # FIXME: changeit
  tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcGdJQkFBS0NBUUVBdkg3Y2=
```

11. Configure kafka-datatopic in **kafka-datatopic_patch.yaml** and in **kustomization.yaml** under **apps/kafka/overlays/env**.

kustomization.yaml

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization



bases:
  - ../aws-managed


patchesJson6902:
- target:
    name: user.data-0
    group: kafka.strimzi.io
    version: v1beta2
    kind: KafkaTopic
  path: kafka-datatopic_patch.yaml


- target:
    name: user.data-1
    group: kafka.strimzi.io
    version: v1beta2
    kind: KafkaTopic
  path: kafka-datatopic_patch.yaml


- target:
    name: user.data-2
    group: kafka.strimzi.io
    version: v1beta2
    kind: KafkaTopic
  path: kafka-datatopic_patch.yaml


- target:
    name: user.data-3
    group: kafka.strimzi.io
    version: v1beta2
    kind: KafkaTopic
  path: kafka-datatopic_patch.yaml


- target:
    name: user.data-4
    group: kafka.strimzi.io
    version: v1beta2
    kind: KafkaTopic
  path: kafka-datatopic_patch.yaml
```

kafka-datatopic_patch.yaml:

```
- op: replace
  path: /spec/partitions
  value: 1
```

12. Configure notification-service **service_patch.yaml**. Add annotation for "service.minervanetworks.com~1url" in **apps/notification-service /overlays/env/service_patch.yaml** with the DNS of the first ingress EKS node.

Example:

```
- op: replace
  path: "/metadata/annotations/service.minervanetworks.com~1url"
  value: https://ingress01.m11.yourtvnow.tv:8090/
```

13. (OPTIONAL) Configure the **statefulset-patch-env-vars.yaml** file in case DRM proxy is required to connect an external DRM service to the Notification Service. For more details, refer to the *Minerva 11 System Notification Service Configuration Guide*, section *Configuring DRM Proxy*.
14. Set ingester bucket name in **deployment-patch.yaml** under **apps/oracle-sync-service/overlays/env**.

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: oracle-sync-service
spec:
  template:
    spec:
      containers:
        - name: oracle-sync-service
          env:
          # S3 settings
            - name: S3_BUCKET
              value: m11psg-ingester-bucket
```

15. Set policy-manager OAUTH and ingester-bucket-name in **statefulset_patch.yaml** under **apps/policy-manager/overlays/env** as follows:

```
    - OAUTH_PROVIDER_TYPE in apps/policy-manager/overlays/env/statefulset_patch.yaml. For example AWS_COGNITO

    - OAUTH_CLIENT_ID in apps/policy-manager/overlays/env/statefulset_patch.yaml.

    - OAUTH_CLIENT_SECRET in apps/policy-manager/overlays/env/statefulset_patch.yaml.

    - OAUTH_ID_CLAIM in apps/policy-manager/overlays/env/statefulset_patch.yaml.

    - OAUTH_DISCOVERY_ENDPOINT in apps/policy-manager/overlays/env/statefulset_patch.yaml. It should follows this structure https://cognito-idp..amazonaws.com//.well-known/openid-configuration
```

```
---
##########################################
# Policy Manager statefulset patch
##########################################
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: policy-manager
spec:
  template:
    spec:
      containers:
        - name: policy-manager
          env:
              ### Internal OAuth Provider
            - name: OAUTH_PROVIDER_TYPE
              value: AWS_COGNITO


#             - name: OAUTH_CLIENT_ID
#               value: <OAUTH_CLIENT_ID>


#             - name: OAUTH_CLIENT_SECRET
#               value: <OAUTH_CLIENT_SECRET>


#             - name: OAUTH_ID_CLAIM
#               value: 'cognito:username'

#             - name: OAUTH_DISCOVERY_ENDPOINT
#               value: >-
#                 https://cognito-idp..amazonaws.com//.well-known/openid-configuration
            # S3 settings
            - name: S3_BUCKET
              value: m11psg-ingester-bucket
```

16. Set recording service in **config.properties** under **apps/recording-service/overlays/env**. For more details, refer to the *Minerva 11 System Recording Services Configuration Guide*.
17. Configure user-service as follows:
    - Add annotation for "service.minervanetworks.com~1url" in **service_patch.yaml** under **apps/user-service/overlays/env** pointing to the (ingress host / AWS CloudFront)/users

```
- op: replace
  path: "/metadata/annotations/service.minervanetworks.com~1url"
  value: https://nginx.m11psg.yourtvnow.tv/users/
```

- Add annotations for "service.minervanetworks.com~1url" for each service to each of the user-service pods in **services-statefulstate-nodes.yaml** under **apps/user-service/overlays/env**. The Annotations URLs need to point to the (ingress host / AWS CloudFront)/users /node/<node number>/.

```
apiVersion: v1
kind: Service
metadata:
  name: user-service-0
  labels:
    app: user-service
    app.kubernetes.io/part-of: m11-cluster
  annotations:
     service.minervanetworks.com/url: https://nginx.m11psg.yourtvnow.tv/users/node/0/
spec:
  selector:
    app: user-service
    statefulset.kubernetes.io/pod-name: user-service-0
  ports:
    - name: client
      port: 8080
      targetPort: 8080
      protocol: TCP
    - name: admin
      port: 9090
      targetPort: 9090
      protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  name: user-service-1
  labels:
    app: user-service
    app.kubernetes.io/part-of: m11-cluster
  annotations:
     service.minervanetworks.com/url: https://nginx.m11psg.yourtvnow.tv/users/node/1/
spec:
  selector:
    app: user-service
    statefulset.kubernetes.io/pod-name: user-service-1
  ports:
    - name: client
      port: 8080
      targetPort: 8080
      protocol: TCP
    - name: admin
      port: 9090
      targetPort: 9090
      protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  name: user-service-2
  labels:
    app: user-service
    app.kubernetes.io/part-of: m11-cluster
  annotations:
     service.minervanetworks.com/url: https://nginx.m11psg.yourtvnow.tv/users/node/2/
spec:
  selector:
    app: user-service
    statefulset.kubernetes.io/pod-name: user-service-2
  ports:
    - name: client
      port: 8080
      targetPort: 8080
      protocol: TCP
    - name: admin
      port: 9090
      targetPort: 9090
      protocol: TCP
---
```

18. (OPTIONAL, in case Minerva private AWS ECR Docker registry is not used in the deployments) Customize docker registry image name for all Minerva microservices in **kustomization.yaml** under **app/<corresponding-app>/overlays/env/** for all Minerva microservices which use Minerva private ECR registry.

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization


bases:
  - ../aws-managed


patchesStrategicMerge:
  - deployment-patch.yaml


images:
  - name: 587308539385.dkr.ecr.eu-west-3.amazonaws.com/minerva/ingester-service
    newName: repo.minervanetworks.com:6000/minerva/ingester-service
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization


bases:
  - ../aws-managed


images:
  - name: 587308539385.dkr.ecr.eu-west-3.amazonaws.com/minerva/maui-webapp
    newName: repo.minervanetworks.com:6000/minerva/maui-webapp
```

19. Configure targetRevision for all apps in the ArgoCD app patch file **<coresponidig_app_name>_patch.yaml** under **app/<corresponding-app>/argocd-app/overlays/env**. targetRevision is a combination of the env branch name + /tag. For example, if the env branch is **env/aws-managed/ta/m11-sandbox-k8s** then targetRevision must be configured with **env/aws-managed/ta/m11-sandbox-k8s/tag**.

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: ingester-service
  namespace: argocd
spec:
  source:
    repoURL: https://git-codecommit.eu-west-3.amazonaws.com/v1/repos/m11-deployment-v2
    targetRevision: env/aws-managed/ta/m11-sandbox-k8s/tag
    path: apps/ingester-service/overlays/env
```

For the last two modifications, use bash for loop with sed in order to modify repoURL value and targetRevision value:

```
a=`ls apps`
for i in $a; do sed -i 's|git-codecommit.eu-west-3.amazonaws.com/v1/repos/m11-deployment|git-codecommit.
eu-west-3.amazonaws.com/v1/repos/m11-deployment-v2|g' apps/$i/argocd-app/overlays/env/*.yaml; done


for i in $a; do sed -i 's|targetRevision: MNPSG_aws_strimzi|targetRevision: env/aws-managed/m11psg-k8s
/tag|g' apps/$i/argocd-app/overlays/env/*.yaml; done
```

20. OPTIONAL (in the event Minerva git repo in Amazon git-codecommit is not used directly): Configure repoURL for all apps in ArgoCD app patch file **<coresponidig_app_name>_patch.yaml** under **app/<corresponding-app>/argocd-app/overlays/env**.

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: ingester-service
  namespace: argocd
spec:
  source:
    repoURL: https://git-codecommit.eu-west-3.amazonaws.com/v1/repos/m11-deployment-v2
    targetRevision: env/aws-managed/minerva/m11psg-k8s/tag
    path: apps/ingester-service/overlays/env
```

21. Select the apps to be enabled in **kustomization.yaml** under **master-app/overlays/env**.

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization


bases:
  - ../../base
  - ../../../apps/deployment-shared/argocd-app/overlays/env
  - ../../../apps/content-consumer/argocd-app/overlays/env
  - ../../../apps/content-manager/argocd-app/overlays/env
  # - ../../../apps/elastic-eck/argocd-app/overlays/env
  # - ../../../apps/elasticsearch/argocd-app/overlays/env
  # - ../../../apps/elasticsearch-logs/argocd-app/overlays/env
  - ../../../apps/export-service/argocd-app/overlays/env
  - ../../../apps/external-services-gateway/argocd-app/overlays/env
  - ../../../apps/fluentbit/argocd-app/overlays/env
  - ../../../apps/ingester-service/argocd-app/overlays/env
  - ../../../apps/ingress-nginx-controller/argocd-app/overlays/env
  - ../../../apps/kafka/argocd-app/overlays/env
  # - ../../../apps/local-volume-static-provisioner/argocd-app/overlays/env
  - ../../../apps/maui-webapp/argocd-app/overlays/env
  # - ../../../apps/minerva-monitoring/argocd-app/overlays/env
  # - ../../../apps/nfs-client/argocd-app/overlays/env
  - ../../../apps/nginx-image-resizer/argocd-app/overlays/env
  - ../../../apps/notification-service/argocd-app/overlays/env
  - ../../../apps/oracle-sync-service/argocd-app/overlays/env
  - ../../../apps/policy-engine/argocd-app/overlays/env
  - ../../../apps/policy-manager/argocd-app/overlays/env
  - ../../../apps/recording-service/argocd-app/overlays/env
  - ../../../apps/srcm/argocd-app/overlays/env
  - ../../../apps/static-service/argocd-app/overlays/env
  - ../../../apps/strimzi-operator/argocd-app/overlays/env
  - ../../../apps/user-service/argocd-app/overlays/env
  - ../../../apps/vmx-admin-gateway/argocd-app/overlays/env
```

22. Reconfigure integration_tests_executor jenkinsfile under **pipelines/integration_tests_executor** as follows:
    - Verify that nos secured is set to false:

```
|     nos:
|       # whether NOS is using HTTPS (value true) or HTTP (value false)
|       secured: false
```

    - Configure the Kafka bootstrap_servers for the environment:

```
|     kafka:
|       default:
|         client_config:
|           bootstrap_servers: "b-1.m11psgmsk.b3qk94.c1.kafka.eu-west-1.amazonaws.com:9094,b-2.
m11psgmsk.b3qk94.c1.kafka.eu-west-1.amazonaws.com:9094,b-3.m11psgmsk.b3qk94.c1.kafka.eu-west-1.
amazonaws.com:9094"
|         discover_ssl_config: true
|         topic_secrets:
|           dir: /var/m11/topic-secrets/
```

23. Commit the changes:

```
git add.
git commit -m "Create env branch <name of the branch starting with env/>"
```

24. Push the changes in newly configured branch.

```
git push --set-upstream origin env/aws-managed/<company name>/<eks cluster name>
```

25. Create a deployment environment tag which ArgoCD will track. On each upgrade or configuration change this tag will be overwritten pointing to different git sha, but the tag name will be always the same. Choose one of the options:
    - OPTION 1: Merge the env branch with latest release (branch master).

```
git checkout master
git pull
git checkout env/aws-managed/minerva/m11psg-k8s
git reset --hard origin/env/aws-managed/minerva/m11psg-k8s
git pull
git merge --no-ff master
git tag -f env/aws-managed/minerva/m11psg-k8s/tag
git push --force origin env/aws-managed/minerva/m11psg-k8s/tag
```

- OPTION 2: Merge the env branch with a certain release (Sys tag. Example Sys/1.0.1).

```
git checkout master
git pull
git checkout env/aws-managed/minerva/m11psg-k8s
git reset --hard origin/env/aws-managed/minerva/m11psg-k8s
git pull
git merge --no-ff Sys/1.0.1
git tag -f env/aws-managed/minerva/m11psg-k8s/tag
git push --force origin env/aws-managed/minerva/m11psg-k8s/tag
```

> **Note:**
>
> Each time the env branch is configured (after checkout the env branch), before making the changes, do a git reset --hard origin/<env_branch name>.

Minerva 11 Installation Performed by ArgoCD

The Minerva 11 installation is performed via ArgoCD using git repository **m11-deployment-v2** and respectively the environment tag.

The UI or the AWS Command Line Interface (CLI) can be used. The steps given below are executed using the CLI interface.

1. Log in to ArgoCD via CLI:

```
argocd login argocdURL --username admin --password xxx --insecure
```

2. Add git repository credentials for **m11-reployment-v2**:

```
argocd repocreds add https://git-codecommit.eu-west-3.amazonaws.com/v1/repos/m11-deployment-v2 --
username  mn_repo-at-xxxxxxxx --password wWVn/znhN80Isg3GmkXRONAyLADPtEamzv3k7zYvxXX=
```

3. Create M11 apps with ArgoCD by replacing int_aws with the environment branch name.

```
argocd app create m11 --dest-namespace argocd --dest-server https://kubernetes.default.svc --repo https:/
/git-codecommit.eu-west-3.amazonaws.com/v1/repos/m11-deployment-v2 --path master-app/overlays/env --
revision <environment-git-tag-previously-created>
```

Example:

```
argocd app create m11 --dest-namespace argocd --dest-server https://kubernetes.default.svc --repo https:/
/git-codecommit.eu-west-3.amazonaws.com/v1/repos/m11-deployment-v2 --path master-app/overlays/env --
revision env/aws-managed/ta/m11-sandbox-k8s/tag
```

4. Open ArgoCD UI and navigate to master-app(m11).
5. As ArgoCD UI will show that all m11 are out of sync, click on SYNC and wait.

Change Log

Revision	Date	Change Details

00	2023-08-01	Initial release. | | **Description** | | 1 | No changes to this version of the document. |