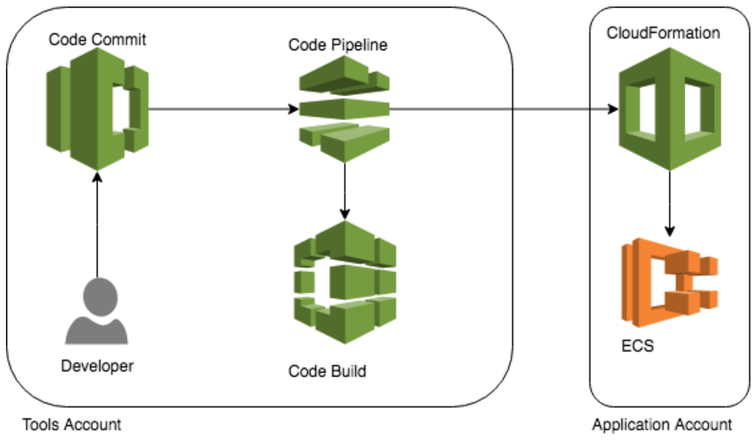
Overview

In this lab we will deploy an Application on ECS using Fargate with CI/CD using Code Pipeline in Control Tower environment.

Creating Accounts using Control Tower Account Factory

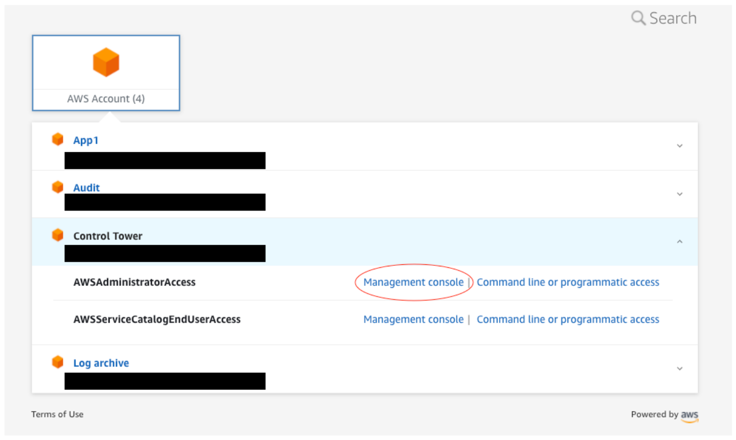
We will create a multi account setup in Control Tower as mentioned below:

A. Tools Account
   Used for Container repositories, Code Commit and setting up Code pipeline to build, push and deploy new images.
B. Application Account
   Account where application will be deployed.

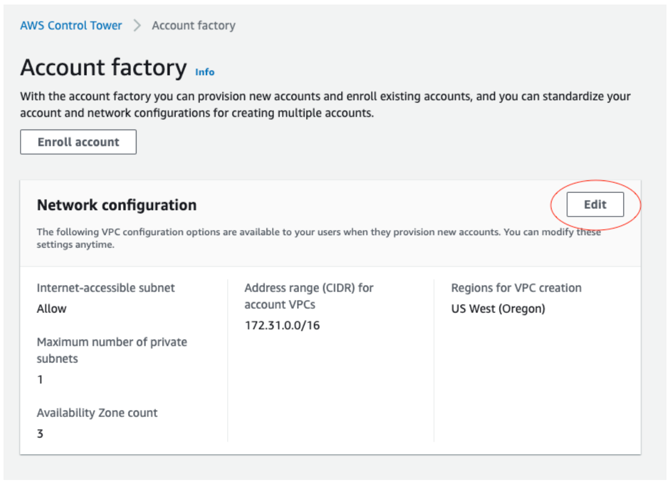


For ECS deployment we'll need to create VPC with public subnet. So before creating the application account using Control Tower Account Factory, we need to update the network configuration.

1. Login to Control Tower account with **AWSAdministratorAccess** role from SSO portal.


Search
AWS Account (4)
App1
[redacted]
Audit
[redacted]
Control Tower
[redacted]
AWSAdministratorAccess
Management console | Command line or programmatic access
AWSServiceCatalogEndUserAccess
Management console | Command line or programmatic access
Log archive
[redacted]
Terms of Use
Powered by aws

2. Go to Control Tower service in the region where it's deployed and Click on Account Factory and Edit network configuration.



AWS Control Tower > Account factory

Account factory Info

With the account factory you can provision new accounts and enroll existing accounts, and you can standardize your account and network configurations for creating multiple accounts.

Enroll account

Network configuration

Edit

The following VPC configuration options are available to your users when they provision new accounts. You can modify these settings anytime.

Internet-accessible subnet: Allow

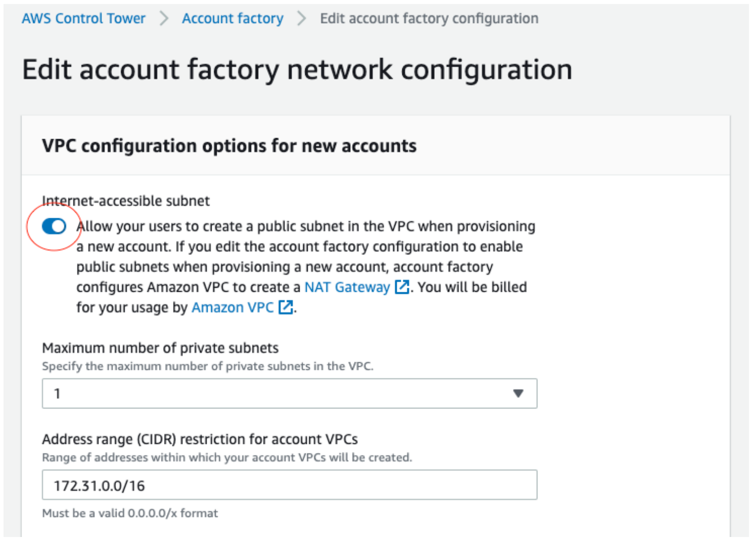
Maximum number of private subnets: 1

Availability Zone count: 3

Address range (CIDR) for account VPCs: 172.31.0.0/16

Regions for VPC creation: US West (Oregon)

3. Toggle the Internet-accessible subnet switch to allow public subnets and click Save.



AWS Control Tower > Account factory > Edit account factory configuration

Edit account factory network configuration

VPC configuration options for new accounts

Internet-accessible subnet

Allow your users to create a public subnet in the VPC when provisioning a new account. If you edit the account factory configuration to enable public subnets when provisioning a new account, account factory configures Amazon VPC to create a NAT Gateway. You will be billed for your usage by Amazon VPC.

Maximum number of private subnets

Specify the maximum number of private subnets in the VPC.

1

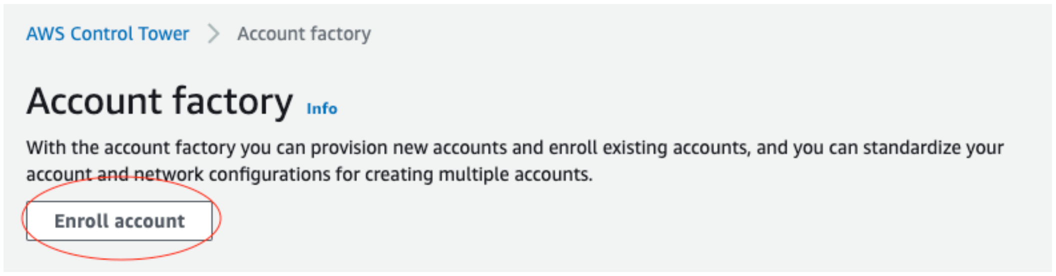
Address range (CIDR) restriction for account VPCs

Range of addresses within which your account VPCs will be created.

172.31.0.0/16

Must be a valid 0.0.0.0/x format

4. Now we are ready to create an Application Account. Click **Enroll account** from **Account factory** console.



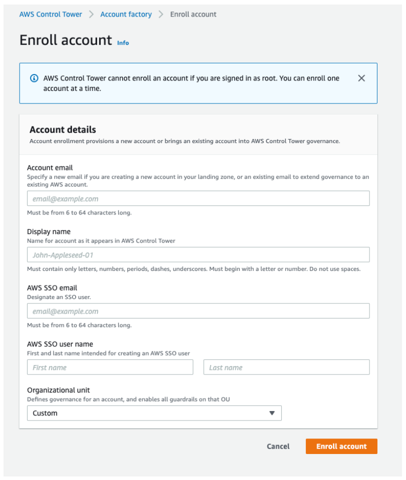
AWS Control Tower > Account factory

Account factory Info

With the account factory you can provision new accounts and enroll existing accounts, and you can standardize your account and network configurations for creating multiple accounts.

Enroll account

5. Fill in the Account details for the new account and click **Enroll account**.
   PS: Choose unique email ID for Account email. For this lab purpose, use AWS SSO email same as Control tower account email, so by default administrator access is added for control tower admin sso user.



AWS Control Tower > Account factory > Enroll account

Enroll account Info

AWS Control Tower cannot enroll an account if you are signed in as root. You can enroll one account at a time.

Account details
Account enrollment provisions a new account or brings an existing account into AWS Control Tower governance.

Account email
Specify a new email if you are creating a new account in your landing zone, or an existing email to extend governance to an existing AWS account.
email@example.com
Must be from 6 to 64 characters long.

Display name
Name for account as it appears in AWS Control Tower
John-Appleseed-01
Must contain only letters, numbers, periods, dashes, underscores. Must begin with a letter or number. Do not use spaces.

AWS SSO email
Designate an SSO user.
email@example.com
Must be from 6 to 64 characters long.

AWS SSO user name
First and last name intended for creating an AWS SSO user
First name | Last name

Organizational unit
Defines governance for an account, and enables all guardrails on that OU
Custom

Cancel | Enroll account

   The status of the account enrollment can be monitored from the AWS Service Catalog dashboard, from the **Provisioned products** list. The account will also show up under accounts in Control tower dashboard. On successful enrollment, the status for the product in Service catalog becomes available and account state in Control tower changes to Enrolled.

6. Next, if you need to create a Tools account then repeat steps 4 and for new account. You need to wait until service catalog product status is changed to available for previous account before enrolling new account.
7. Identify the Account id for Application and Tools account from Control tower accounts section, we'll use these account numbers later.

Cross account prerequisite

Services used in Tools Account:

A. Code Commit, Code Pipeline, Code Build
B. S3 bucket used by Code Pipeline
C. ECR repositories

Services used in Application Account:

A. CloudFormation (Used by code pipeline to deploy application)
B. ECS
C. KMS key for Code Pipeline artifact Bucket

As Code Pipeline uses S3 bucket with encryption to store artifacts, we need to update the IAM role used by code pipeline with permission to the KMS key used. And an IAM role in application account which can be consumed by code pipeline to deploy app.
For this cross-account setup, IAM roles with cross account trust for used resources are required.
So as prerequisite, first we'll create KMS key to use with code pipeline and IAM roles in application account.

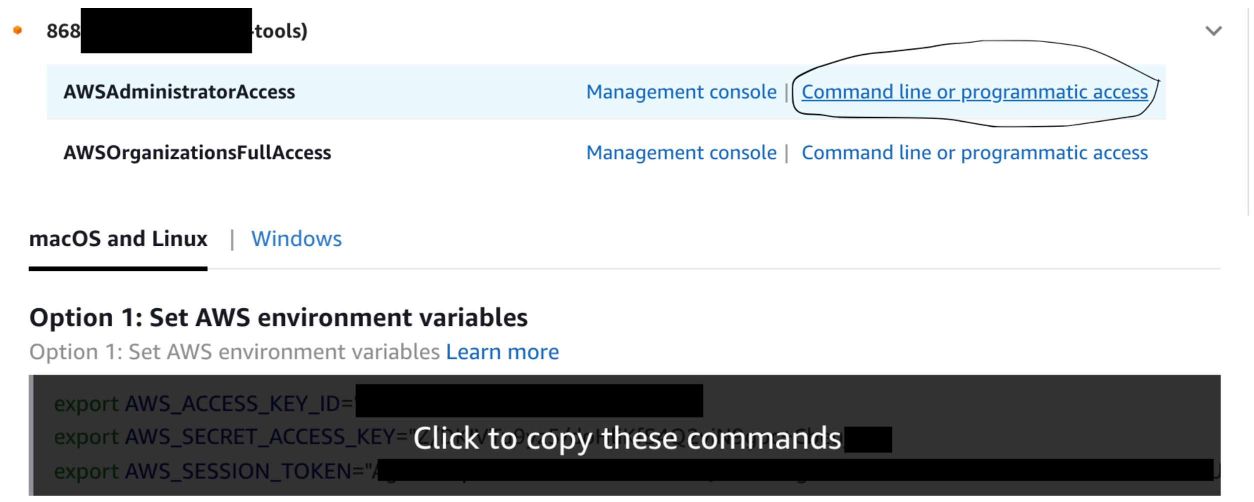
Follow below instructions to create CloudFormation stack set instance in tools and application account to create KMS, S3 bucket and IAM.

8. Clone code repo https://github.com/aws-samples/aws-control-tower-managed-ecs-deployments
9. Login to Control tower account with **AWSAdministratorAccess** role from SSO portal
10. Navigate to the **StackSet** console under CloudFormation in the **control tower** account
11. Click "**Create StackSet**"
12. Select "**Upload a template to Amazon S3**" and select pre_req.yaml template under ct-lab folder from cloned repo in step 8.
13. Give the **StackSet** a name: ECSPreReq.
14. Enter tools account number (identified in step 7) where pipeline for ECS deployment will be created for **ToolsAccount** parameter.
15. Click Next
16. Select **Self service permissions** for Permission option.
17. Select **AWSControlTowerStackSetRole** from the list under IAM Admin Role ARN. Type **AWSControlTowerExecution** for IAM Execution Role Name (change the default value) and click on Next.
18. Enter Application account number (identified in step 7) in "Deploy stacks in accounts" value.
19. Specify the primary region where you launched Control Tower service.
20. Leave rest as default and click Next
21. Acknowledge the **Capabilities** and click **Submit.**
22. Wait for the stack instance to be created successfully. Status for the stack instance should be **CURRENT**. Copy the stack id.
23. This template created below resource in Application account:
    a. CodePipelineRole to be assumed by code pipeline from tools account
    a. CloudFormationExecutionRole to execute CF in application account
    b. KMS key and alias
24. Login to Application account from SSO portal
25. Go to CloudFormation service and search for the stack with id copied in step 22. Get the output value for **CMKArn** from recently deployed CloudFormation stack, for later use.

Create Repositories

We'll need to create Code commit repository to push application code and ECR repository to push docker images for application in Tools Account (identified in step 7)

26. Setup programmatic access on terminal for tools account from SSO portal.



868[redacted]tools)

AWSAdministratorAccess   Management console | Command line or programmatic access

AWSOrganizationsFullAccess   Management console | Command line or programmatic access

macOS and Linux | Windows

Option 1: Set AWS environment variables

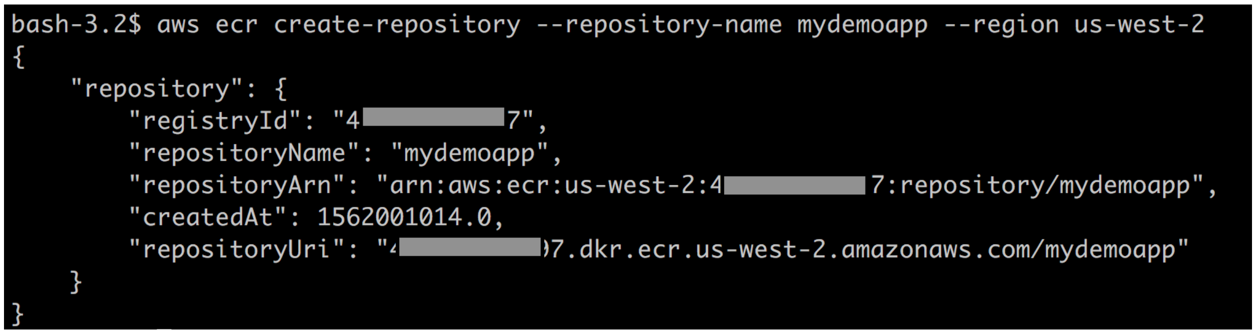
Option 1: Set AWS environment variables Learn more

```
export AWS_ACCESS_KEY_ID=[redacted]
export AWS_SECRET_ACCESS_KEY=[redacted]
export AWS_SESSION_TOKEN="[redacted]
```

Click to copy these commands

27. Paste copied commands from above step into terminal.
28. Create ECR repository, run below command. Pick the region same as Control Tower deployment and repo name (i.e. mydemoapp, in lower case without spaces only)

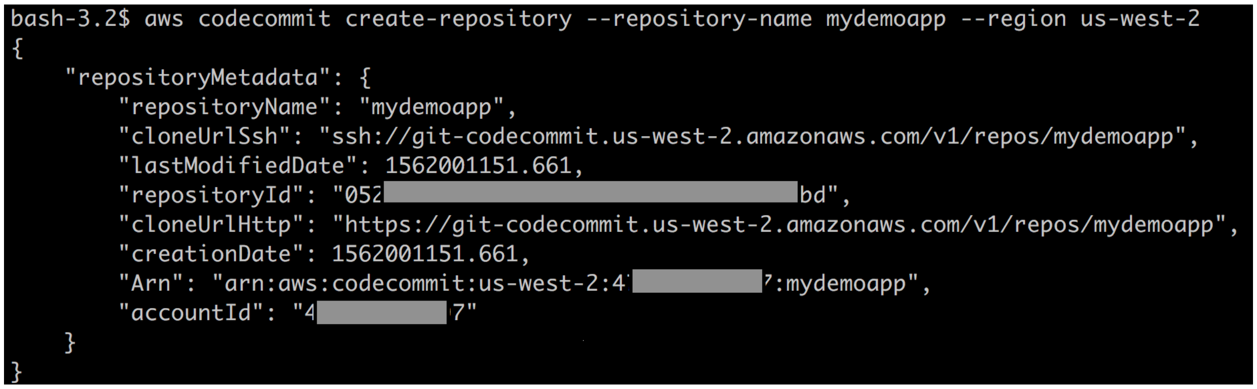
```
aws ecr create-repository --repository-name <Name_Of_Repo> --region
<same as CT region>
```



```
bash-3.2$ aws ecr create-repository --repository-name mydemoapp --region us-west-2
{
    "repository": {
        "registryId": "4[redacted]7",
        "repositoryName": "mydemoapp",
        "repositoryArn": "arn:aws:ecr:us-west-2:4[redacted]7:repository/mydemoapp",
        "createdAt": 1562001014.0,
        "repositoryUri": "[redacted]7.dkr.ecr.us-west-2.amazonaws.com/mydemoapp"
    }
}
```

29. Create Code Commit repo, run below command. the region same as Control Tower deployment and repo name (i.e. mydemoapp, in lower case without spaces only)

```
aws codecommit create-repository --repository-name < Name_Of_Repo > --
region <same as CT region>
```



```
bash-3.2$ aws codecommit create-repository --repository-name mydemoapp --region us-west-2
{
    "repositoryMetadata": {
        "repositoryName": "mydemoapp",
        "cloneUrlSsh": "ssh://git-codecommit.us-west-2.amazonaws.com/v1/repos/mydemoapp",
        "lastModifiedDate": 1562001151.661,
        "repositoryId": "052[redacted]bd",
        "cloneUrlHttp": "https://git-codecommit.us-west-2.amazonaws.com/v1/repos/mydemoapp",
        "creationDate": 1562001151.661,
        "Arn": "arn:aws:codecommit:us-west-2:4[redacted]7:mydemoapp",
        "accountId": "4[redacted]7"
    }
}
```

30. Copy the cloneUrlHttp value from above command's output.
31. Before we are able to push code to our CodeCommit repo, we will need to configure our credentials allow HTTP, run below command

```
git config --global credential.helper '!aws codecommit credential-
helper $@'

git config --global credential.UseHttpPath true
```

32. Add code commit remote to repo directory

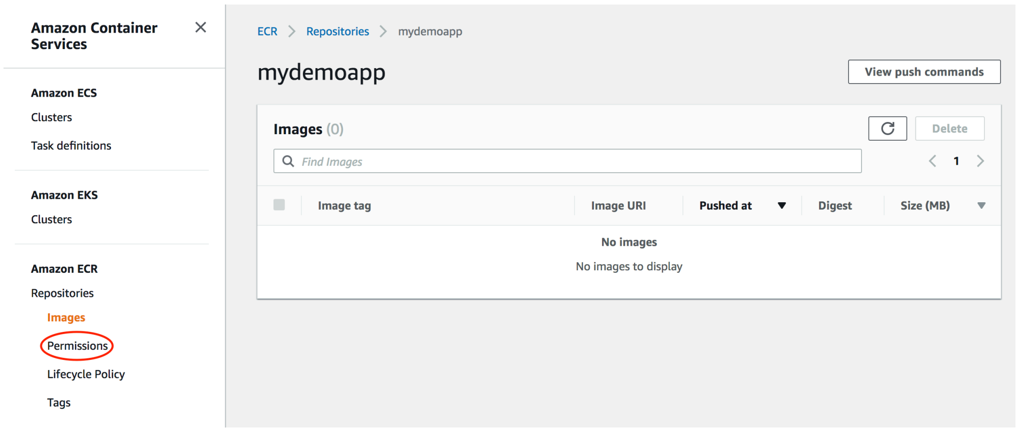
```
git remote add codecommit < cloneUrlHttp copied earlier in step 30>
```

33. Push code to code commit

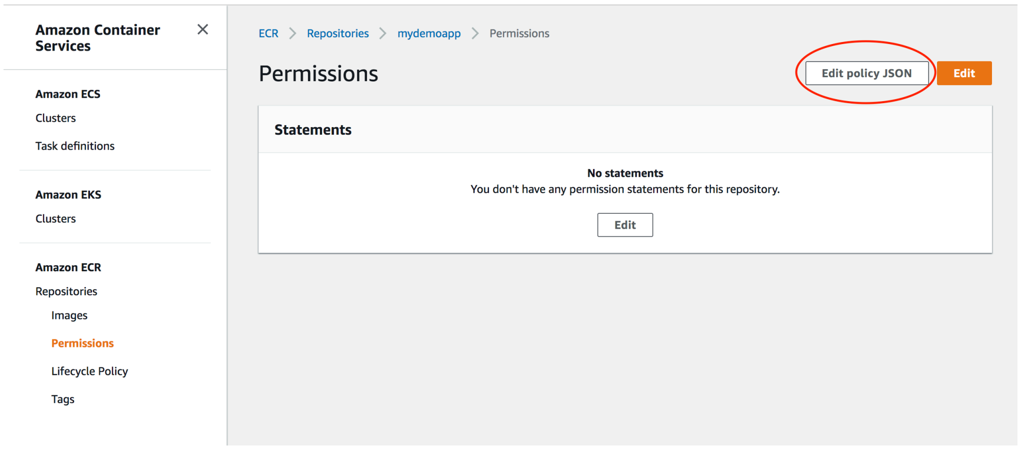
```
git push codecommit master
```

Update ECR repository permission to allow Application Account to pull docker images

34. Login to Tools account from SSO portal
35. Go to ECR service, select the repository created in step 36.
36. Click on permissions on left side menu.



37. Click on Edit policy JSON, on right top corner.



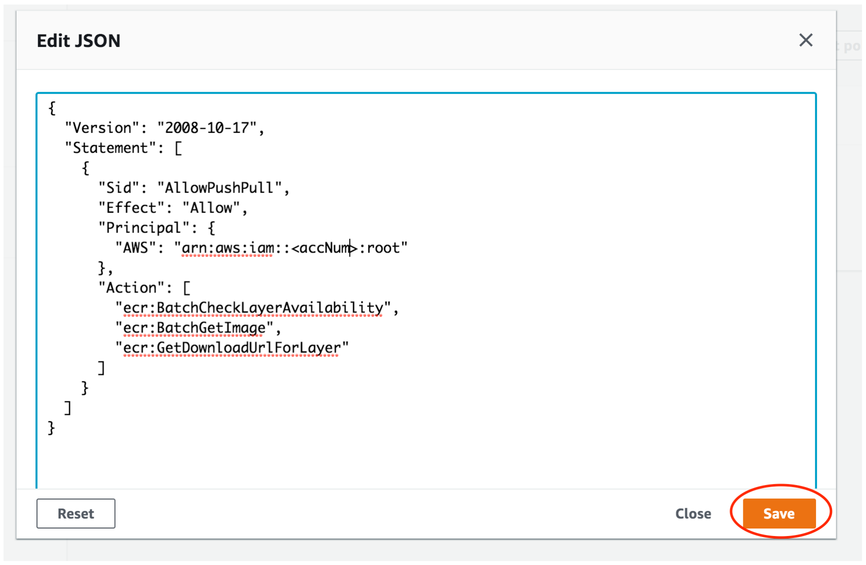
Permissions

Statements

No statements
You don't have any permission statements for this repository.

38. Replace the account number of Application account and paste this JSON into policy document and click save.

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowPull",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<ApplicationAccountNum>:root"
      },
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ]
    }
  ]
}
```

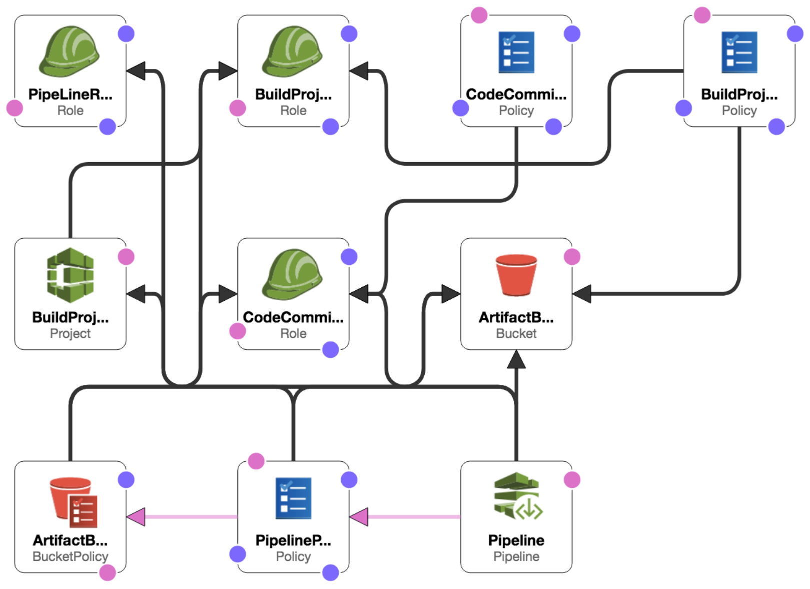


```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowPushPull",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<accNum>:root"
      },
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ]
    }
  ]
}
```

Deploy Pipeline

Now we'll deploy the code pipeline and resources required in tools account.

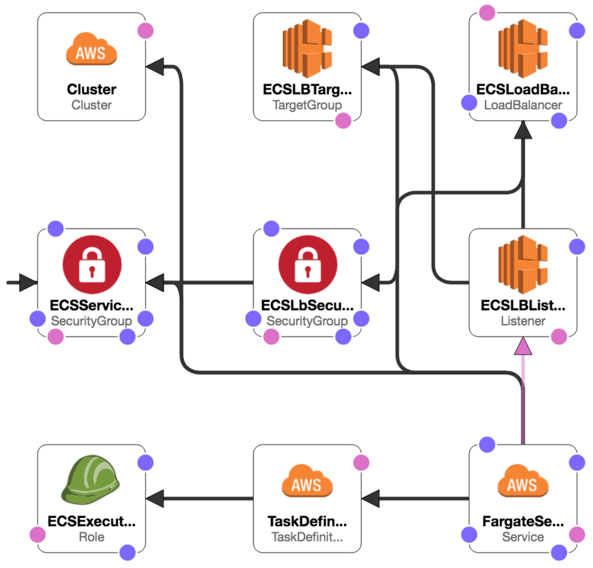
39. Login to Application Account from SSO portal
40. Go to VPC, copy public subnet ids and vpc id
41. Login to Control tower account from SSO portal
42. Navigate to the **StackSet** console under CloudFormation in the **control tower** account
43. Click "**Create StackSet**"
44. Select "**Upload a template to Amazon S3**" and select pipeline.yaml template under ct-lab folder from repo cloned in step 8.
45. Give the **StackSet** a name: ECSPipeline.
46. For **CMKARN** parameter enter KMS key ARN copied earlier in step 25.
47. Enter account number for application identified in step 7, where ECS will be deployed, for **ApplicationAccount** parameter.
48. Enter application name (i.e. mydemoapp, in lower case without spaces only) for **ApplicationName** parameter.
49. Enter code commit repository name, same as used in step 29, for **CodeCommitRepositoryName** parameter.
50. Enter ECR repository name, same as used in step 28, for **ECRRepositoryName** parameter.
51. Enter comma separated (no spaces) subnet ids of public subnets from Application Account, copied in step 40, for **PublicSubnets** parameter.
52. Enter VPC id from Application Account, copied in step 40, for **VpcId** parameter.
53. Click Next
54. Select **Self service permissions** for Permission option.
55. Select **AWSControlTowerStackSetRole** from the list under IAM Admin Role ARN. Type **AWSControlTowerExecution** for IAM Execution Role Name (change the default value) and click on Next.
56. Enter only tools account number (identified in step 7) in "Deploy stacks in accounts" value.
57. Use the primary region where you launched Control Tower service.
58. Leave rest as default and click Next.
59. Acknowledge the **Capabilities** and click **Submit.**
60. Wait for the stack instance to be created successfully. Status for the stack instance should be **CURRENT**.
61. This template created below resource in tools accounts:
    a. Code Pipeline Role and policy
    c. Code Build Role and policy
    d. Code Commit Role and Policy
    e. S3 bucket for Pipeline and bucket policy
    f. Code Pipeline and Code Build project



What will code pipeline do?

1. The pipeline checks out code from the AWS Code Commit repository to S3.
2. Then AWS Code Build will decrypt artifacts in the S3 bucket and run build steps.
3. Code build will build new docker image and push to ECR repositories.
4. Then Code build will generate a parameter JSON file with newly built images URI, which will be used with CloudFormation to deploy/update ECS resources.
   (Look at buildspec.yml file for steps executed by Code Build)
5. Once code build has finished successfully, code pipeline assumes IAM of application account and deploy CloudFormation template to application account.
6. As part of ECS CloudFormation deployment below mentioned resources are created for given application:
   a. ECS Cluster
   b. Task Definition
   c. Application Load Balancer
   d. Security groups for ALB and ECS Service
   e. ECS Fargate Service

   (Look at ecs.yml file for resources being created)



To verify the pipeline, go to tools account and CodePipeline service. You should find the pipeline created for the application (pipeline name is same as **ApplicationName** parameter used earlier in step 48) and check its execution stages.
Once pipeline is done executing successfully, login to Application account and go to CloudFormation. You should find the cloudformation stack for the application (stack name is same as **ApplicationName** parameter used earlier in step 48). Check the outputs and copy the AppDNSName value and paste into the browser. You should see the deployed application.