aws FOR FINANCIAL SERVICES

Prototyping and Cloud Engineering

FSI PACE

Financial Services Industry

OPA on AWS

Security

Orchestrate Platform and Applications

FSI PACE

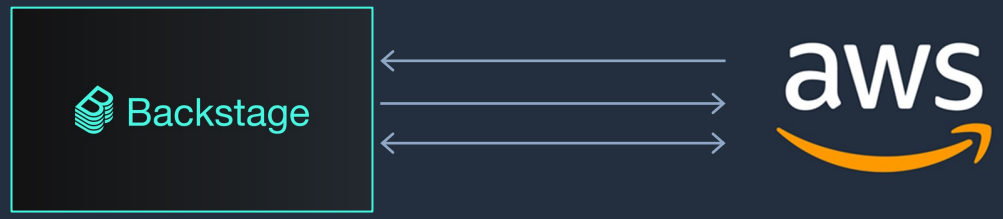
Platform Engineering Team

fsi-pace-pe@amazon.com

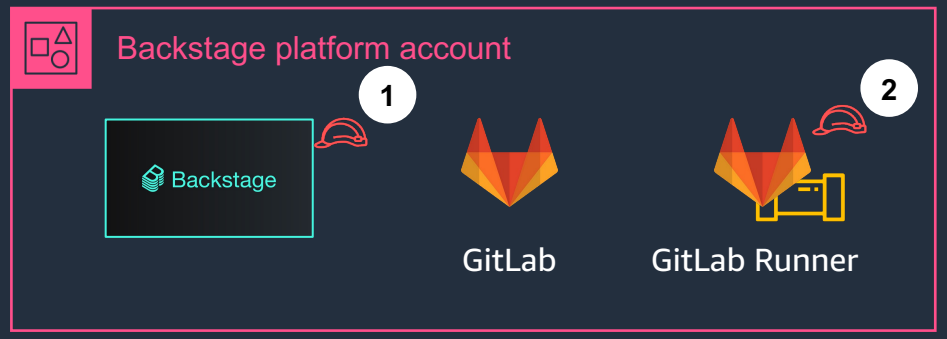
OPA – Orchestrate Platform and Applications

Foundations to build Integrated IDP with AWS

- Source Code
- Designed for Enterprise
- Templates
- Examples
- Backstage Plugins
- Security Best Practices



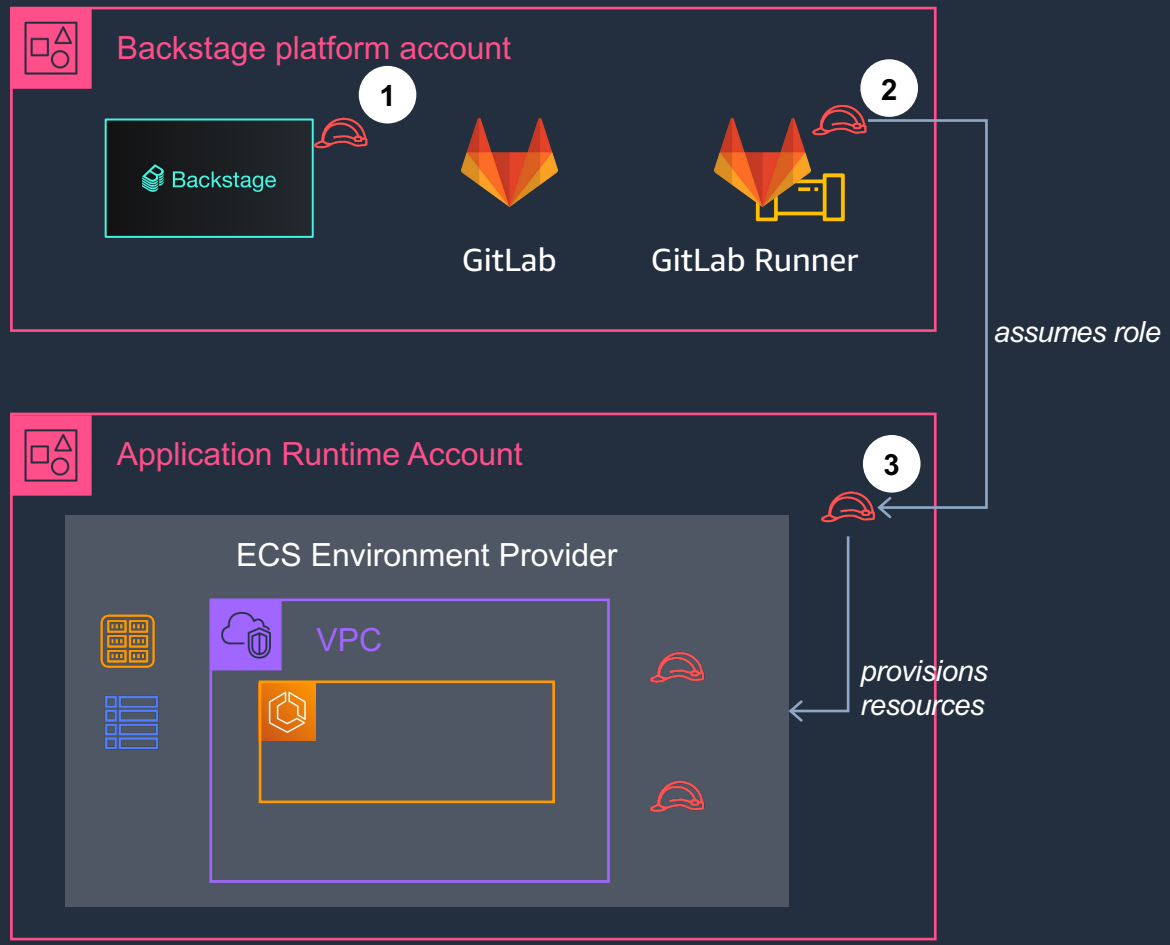
Architecture



1. Backstage platform role (ECS task execution role)

2. GitLab Runner EC2 instance role

Architecture – Environment Provider Provisioning

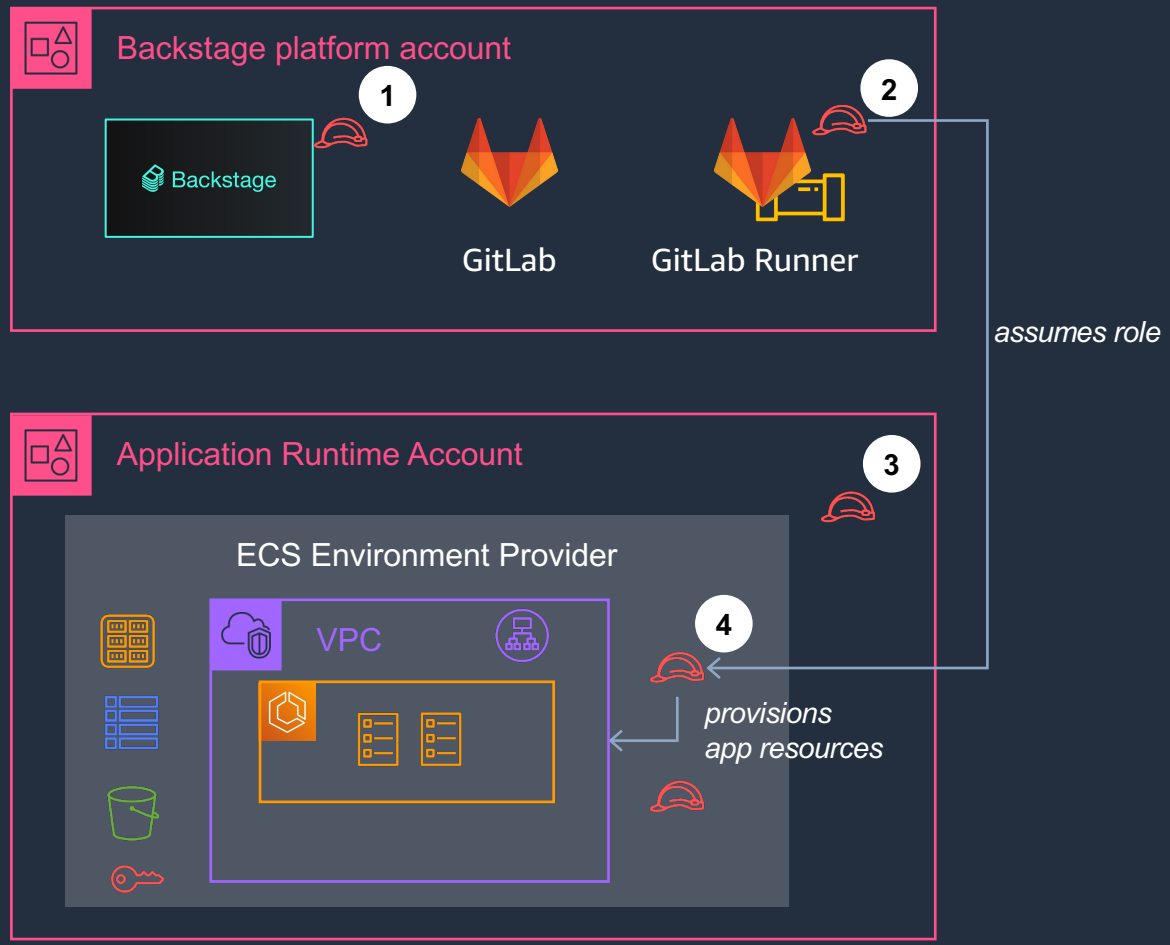


1. Backstage platform role (ECS task execution role)

2. GitLab Runner EC2 instance role

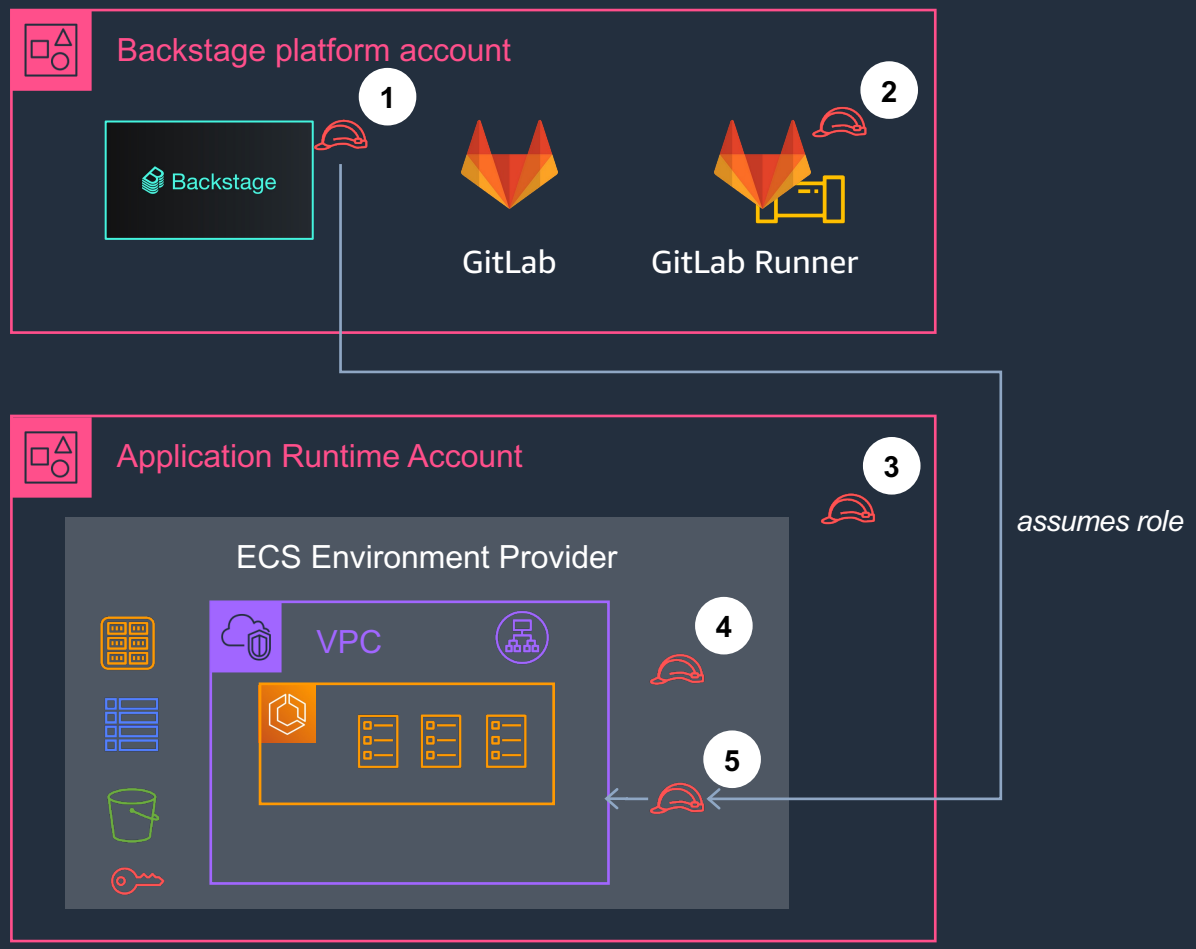
3. Environment provisioning role
   This role is not created by the OPA on AWS solution.
   It must be created manually and allow the GitLab Runner instance role the sts:AssumeRole permission in its trust policy

Architecture – Application Provisioning



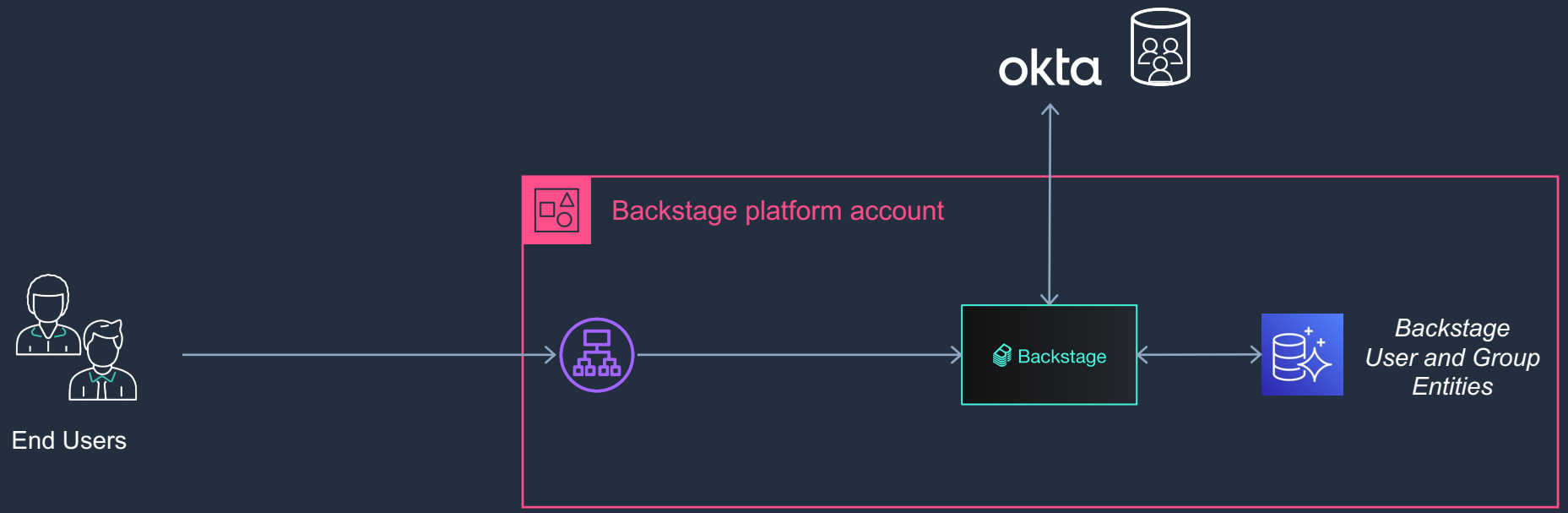
1. Backstage platform role (ECS task execution role)
2. GitLab Runner EC2 instance role
3. Environment provisioning role
   This role is not created by the OPA on AWS solution.
   It must be created manually and allow the GitLab Runner instance role the sts:AssumeRole permission in its trust policy
4. Application provisioning role

Architecture – Application Operations



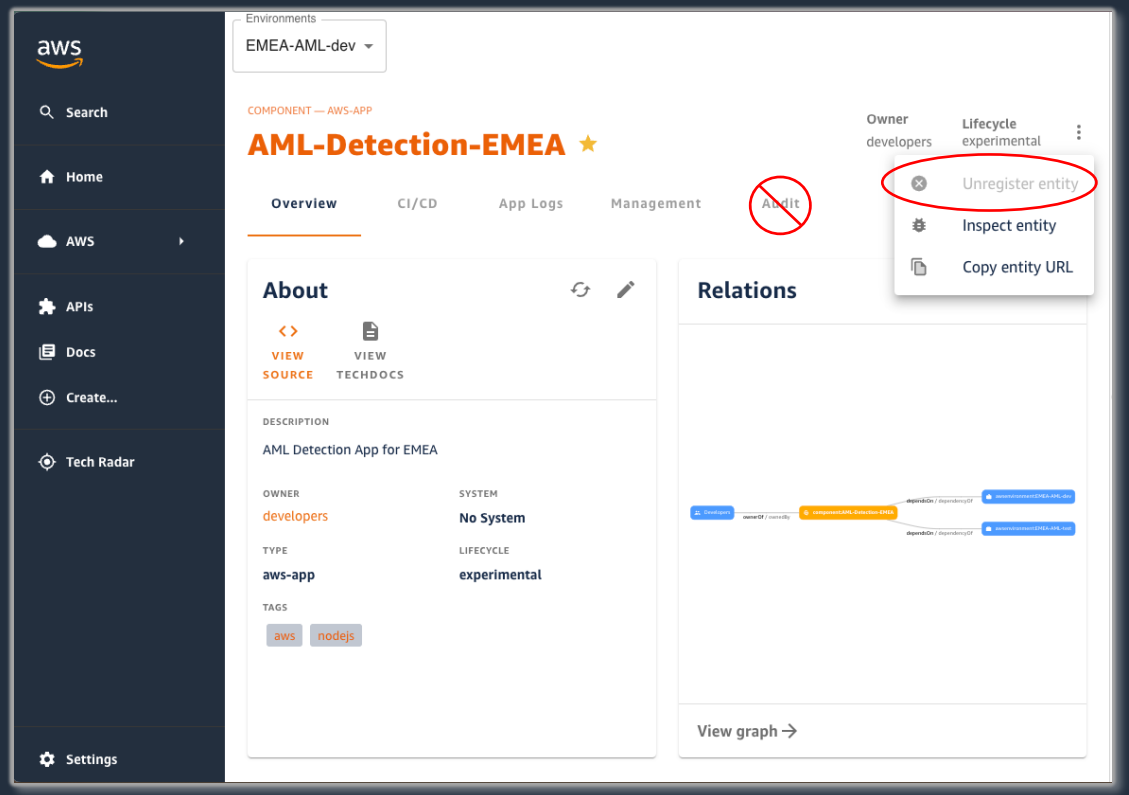
1. Backstage platform role (ECS task execution role)
2. GitLab Runner EC2 instance role
3. Environment provisioning role
   This role is not created by the OPA on AWS solution.
   It must be created manually and allow the GitLab Runner instance role the sts:AssumeRole permission in its trust policy
4. Application provisioning role
5. Application operations role

End user authentication



https://backstage.io/docs/auth/

Permissions



Default Allow for all actions

Write Policies to return policy decisions

Examples:

- Disable UI options
- Hide UI widgets
- Return 401 ‘not authorized’ API responses

https://backstage.io/docs/permissions/overview

Writing a custom policy – OPA example policy decisions

```
// backstage/packages/backend/src/plugins/OpaSamplePermissionPolicy.ts

// DENY OPA audit read access if the user is in the Testers group
if (isPermission(request.permission, readOpaAppAuditPermission) && ownershipGroups.includes(TESTERS_GROUP)) {
  return { result: AuthorizeResult.DENY };
}

// DENY access to 'aws-environment' or 'aws-environment-provider' templates if the user cannot claim ownership of the
entity
if (isPermission(request.permission, catalogEntityReadPermission)) {
  return createCatalogConditionalDecision(request.permission, {
    not: {
      allOf: [
        catalogConditions.isEntityKind({ kinds: ['template'] }),
        {
          anyOf: [
            catalogConditions.hasSpec({ key: 'type', value: 'aws-environment' }),
            catalogConditions.hasSpec({ key: 'type', value: 'aws-environment-provider' }),
          ],
        },
      ],
    },
  });
}
```

Specify the custom policy

```
// backstage/packages/backend/src/plugins/permission.ts

import { OpaSampleAllowAllPolicy, OpaSamplePermissionPolicy } from './OpaSamplePermissionPolicy';


export default async function createPlugin(env: PluginEnvironment): Promise<Router> {
  return await createRouter({
    config: env.config,
    logger: env.logger,
    discovery: env.discovery,
    // policy: new OpaSampleAllowAllPolicy(),
    policy: new OpaSamplePermissionPolicy(),
    identity: env.identity,
  });
}
```

https://backstage.io/docs/permissions/overview

aws FOR FINANCIAL SERVICES

Thank you!

Orchestrate Platform and Applications

fsi-pace-pe@amazon.com