BRAINLY

GETTING STARTED WITH AWS

The proper way - No more IAM users

May 2024

Agenda

1. About the presenter
2. Introduction
3. Step-by-step setup
4. Adding other accounts and setting CLI
5. Summary

PRESENTER

Krzysztof Szyper

Solutions Architect,

Production Infrastructure team,

Brainly

- Wrote first programs at 6yo with C-64.
- Using AWS since 2012.
- SysOps and DevOps background.
- Everything-as-Code enthusiast.
- Managing low-level cloud infrastructure.
- After work doing some portrait and aviation photography.

#1 AI EDU APP IN THE WORLD

AI Learning Companion^{tm}
15M daily active users
250M+ answers in Knowledge Base

QUICK COMPARISON

IAM USERS

- Simple solution to start with
- Complex to maintain with more than one user or AWS account
- Long-lived keys pose security risk

SAML WITH IAM ROLES

- Not straightforward, needs 3rd party service
- Scales easily with users and accounts
- RBAC out of the box
- Session token live for only few hours

IT CAN BE FAMILIAR

If you’re working in a company with many AWS accounts you may already be using this approach.

IS IT WORTH THE EFFORT

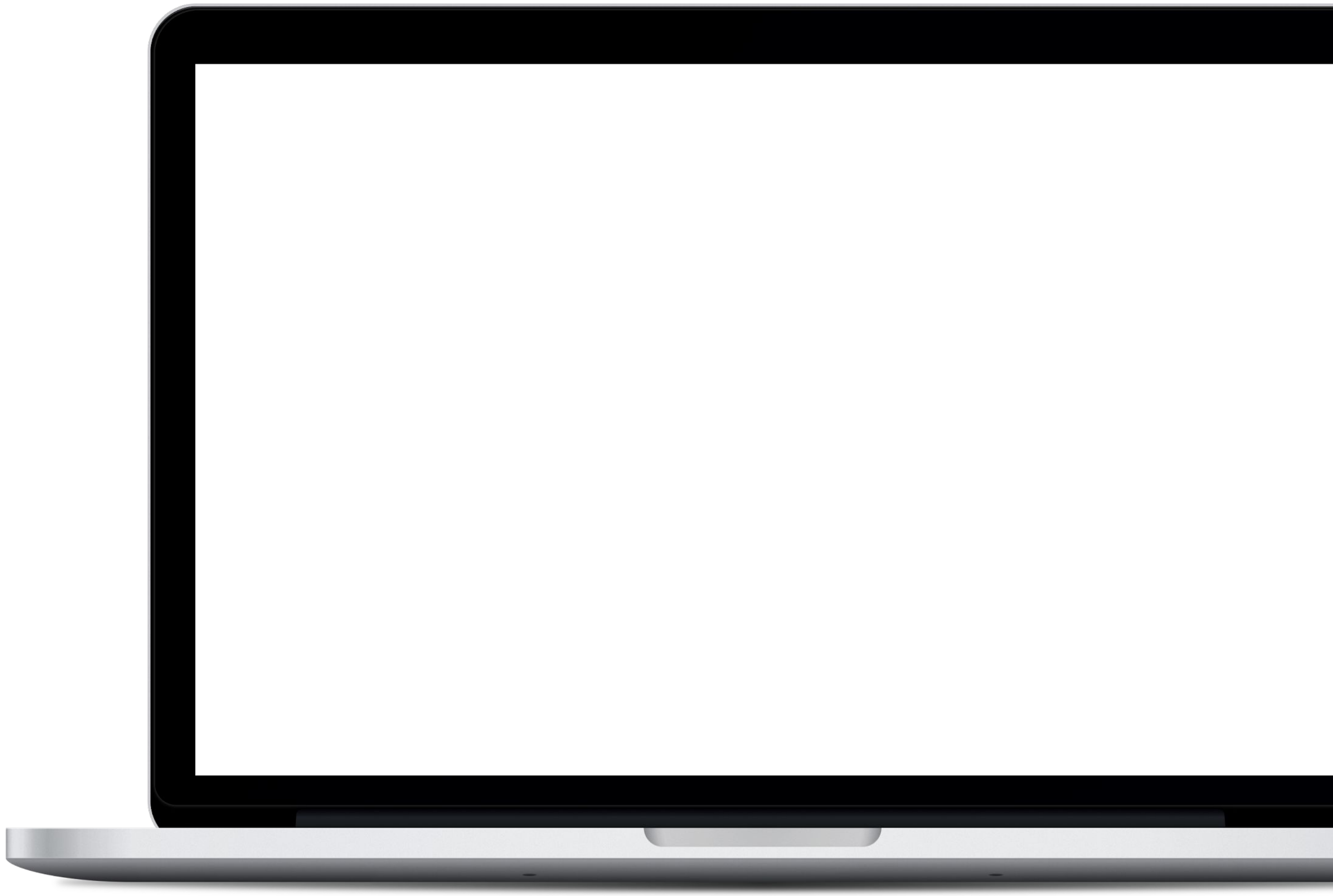
Splitting your personal AWS account to many and using SAML will let you learn about workloads, cross-account access, etc.

Step-by-step guide

This is simplified setup to demonstrate idea and initial setup

PREREQUISITES

What is needed in this example?

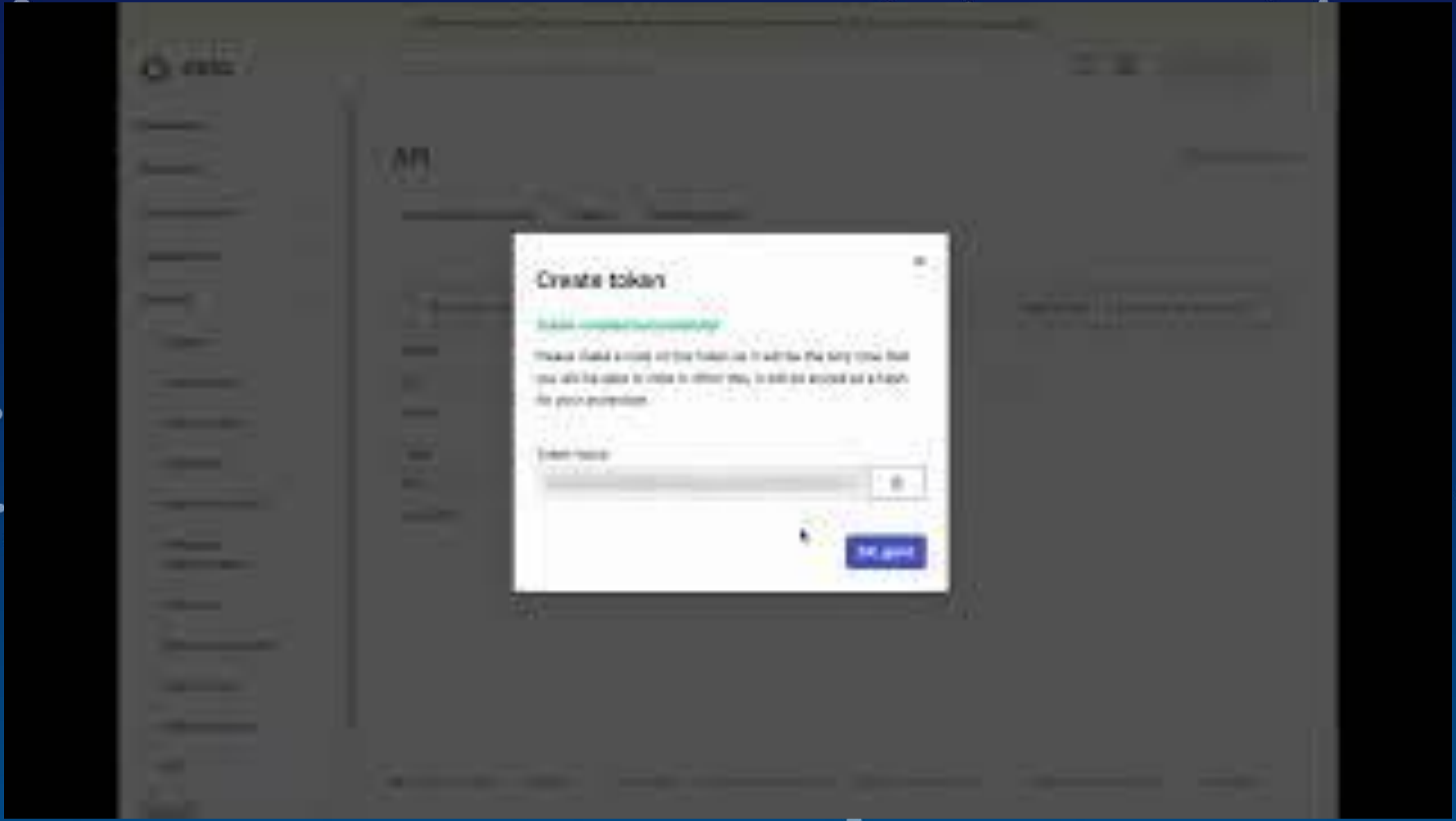


- Unix-like OS
- Terraform
- Okta account
- At least one AWS account

OKTA API TOKEN

Token will be used by Terraform to provision Okta resources.

CREATE OKTA API TOKEN FOR TERRAFORM



Create token

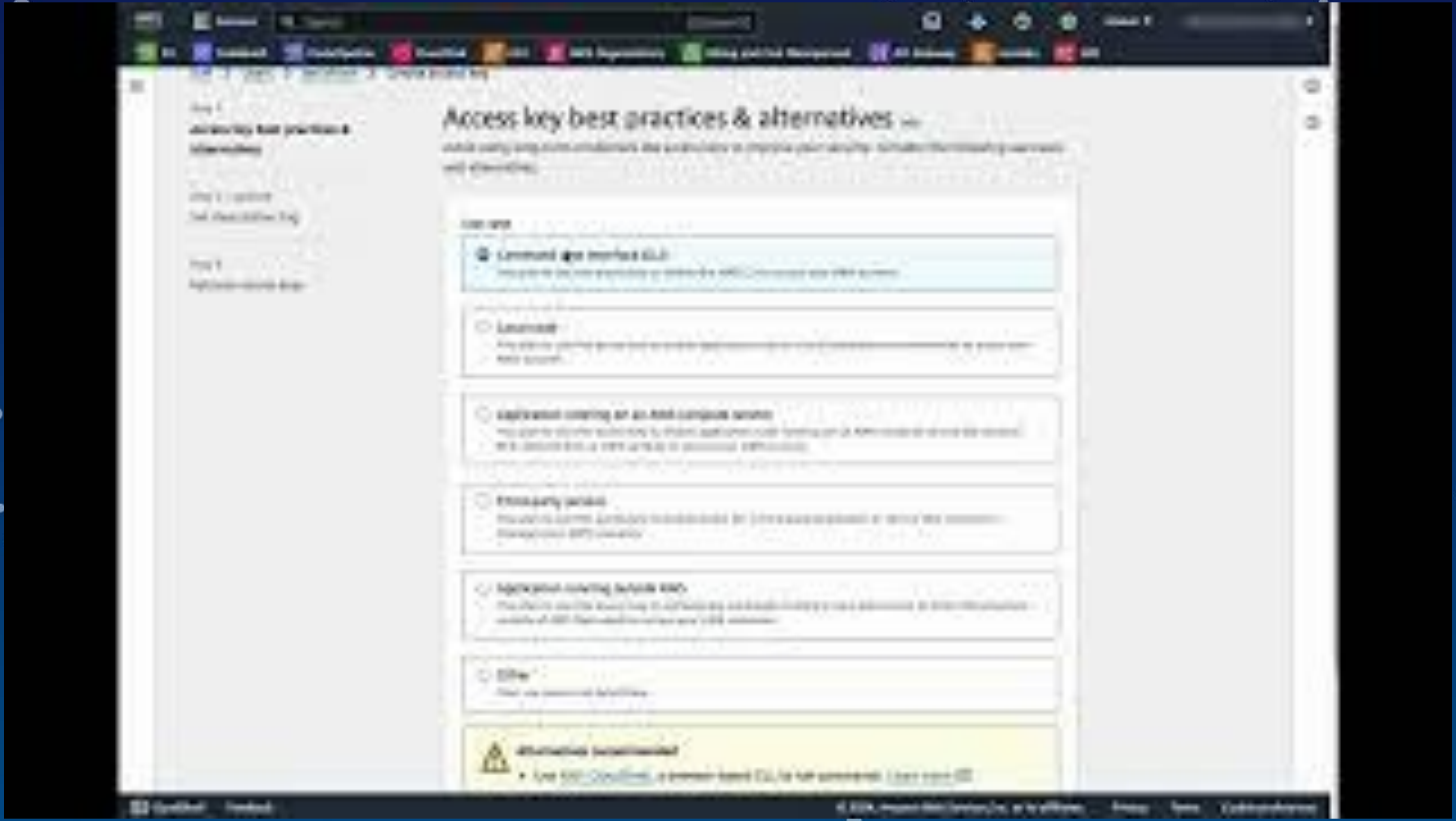
Token created successfully!

Please make a copy of the token as it will be the only time that you will be able to view it. After that, it will be stored as a hash for your protection.

IAM USER FOR TERRAFORM

IAM user for initial provisioning of AWS resources. Can be deleted afterwards.

CREATE AWS IAM USER FOR TERRAFORM



Access key best practices & alternatives

[illegible]

Use case

● Command Line Interface (CLI)
[illegible]

○ Local code
[illegible]

○ Application running on an AWS compute service
[illegible]

○ Third-party service
[illegible]

○ Application running outside AWS
[illegible]

○ Other
[illegible]

⚠ Alternatives recommended
- Use AWS CloudShell, a browser-based CLI, to run commands. Learn more

SET LOCAL ENVIRONMENT

Okta token and AWS credentials for Terraform providers.

SET CREDENTIALS IN TERMINAL

```
$
$ clear
$ export OKTA_API_TOKEN=[redacted]
$ cat ~/.aws/config
[default]
region = us-west-2
output = json
cli_pager =

[illegible]
```

PREPARE TERRAFORM CODE

User access assignment and initial configuration,
e.g. Okta organization.

PREPARE TERRAFORM CODE

```
# [illegible]

terraform {
  required_version = "[illegible]"

  required_providers {
    aws = {
      source  = "[illegible]"
      version = "[illegible]"
    }

    okta = {
      source  = "[illegible]"
      version = "[illegible]"
    }

    [illegible] = {
      source  = "[illegible]"
      version = "[illegible]"
    }
  }
}

# [illegible]
provider "aws" {
  [illegible] = "[illegible]"
  profile = "[illegible]"
  region  = [illegible]
}

# [illegible]
provider "okta" {
  org_name = [illegible]
  base_url = [illegible]
}
```

PROVISION RESOURCES

SAML app and access groups in Okta, IAM user and roles for Okta in AWS.

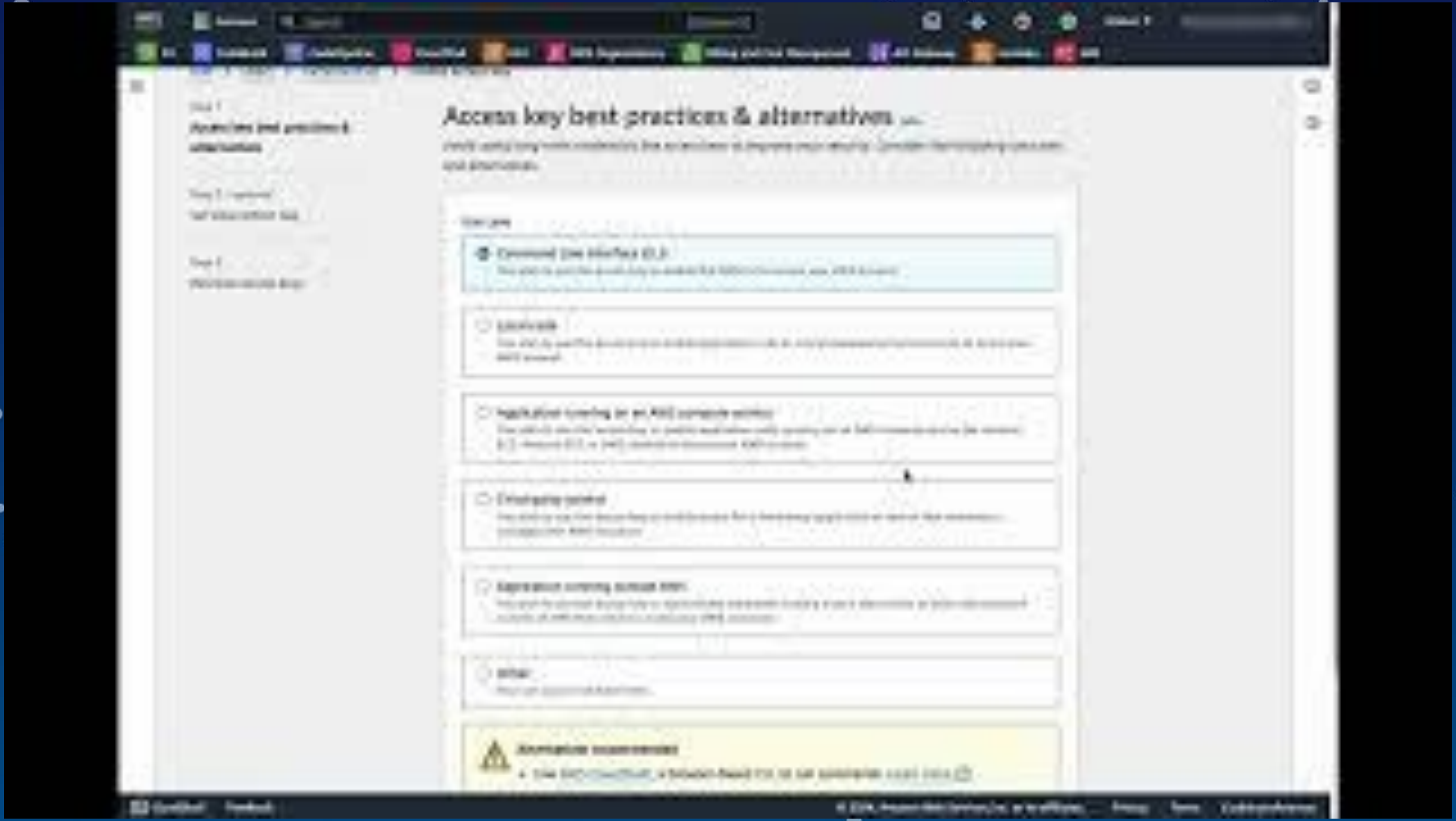
RUN TERRAFORM APPLY

```
$ export OKTA_API_TOKEN=[redacted]
$ export TF_VAR_[illegible]
$ terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of [illegible] from the dependency lock file
- Reusing previous version of okta/okta from the dependency lock file
- Using previously-installed hashicorp/aws [illegible]
- Using previously-installed [illegible]
- Using previously-installed okta/okta [illegible]

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
$ terraform plan
█
```

IAM ACCESS KEYS FOR OKTA

Keys will be used to read IAM roles and assume them in all AWS accounts.

GET KEYS FOR OKTA IAM USER



Step 1
Access key best practices & alternatives

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

Access key best practices & alternatives

[illegible]

Use case

● Command Line Interface (CLI)
[illegible]

○ Local code
[illegible]

○ Application running on an AWS compute service
[illegible]

○ Third-party service
[illegible]

○ Application running outside AWS
[illegible]

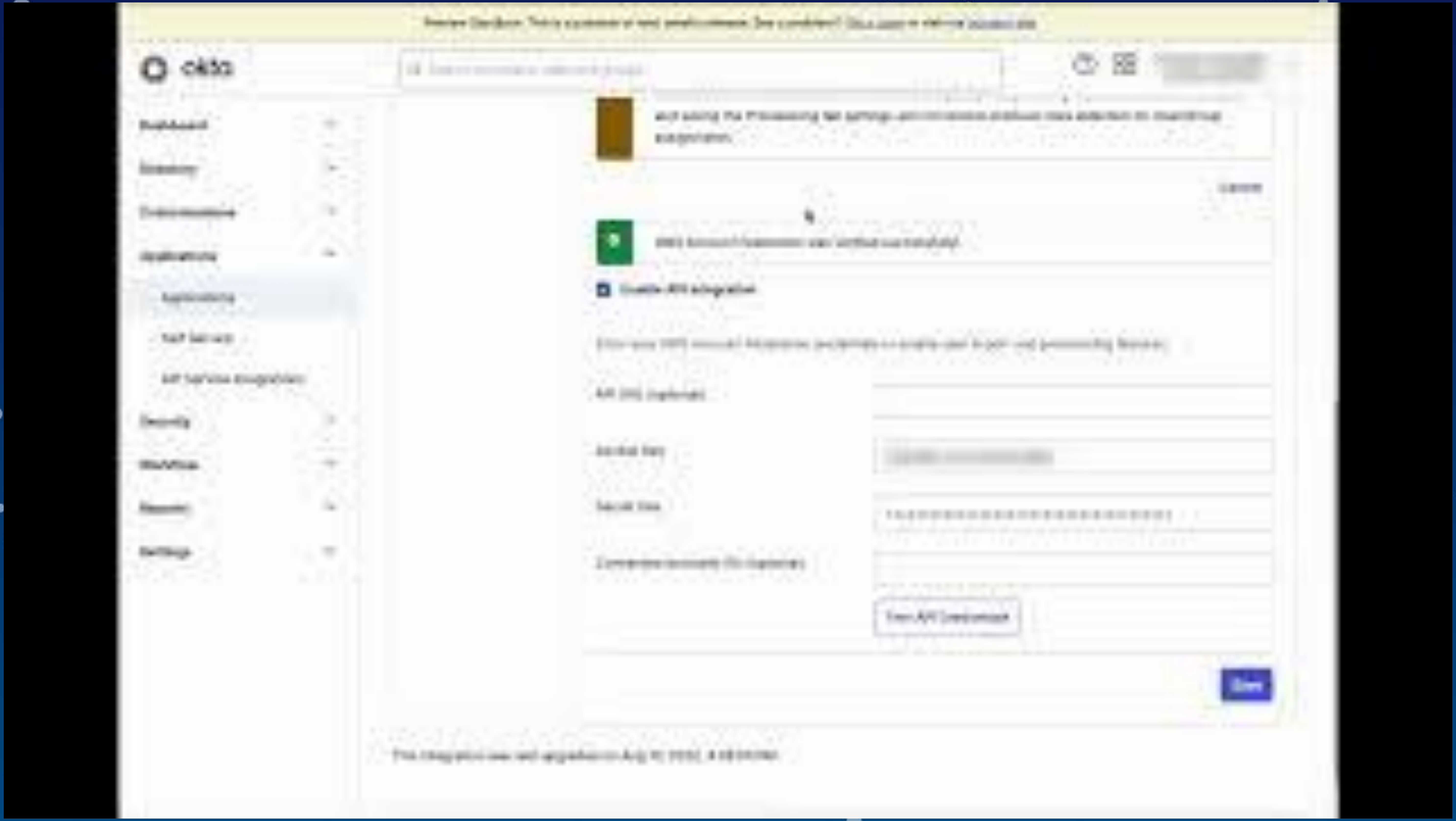
○ Other
[illegible]

⚠ Alternative recommended
- Use AWS CloudShell, a browser-based CLI, to run commands. [illegible]

CONFIGURE OKTA AND TEST LOGIN

Enter those IAM access keys in Okta provisioning configuration.

UPDATE OKTA PROVISIONING



[illegible]

AWS Account Federation was verified successfully!

☑ Enable API integration

[illegible]

API URL (optional)

Access Key [redacted]

Secret Key

Connected Accounts IDs (optional)

Test API Credentials

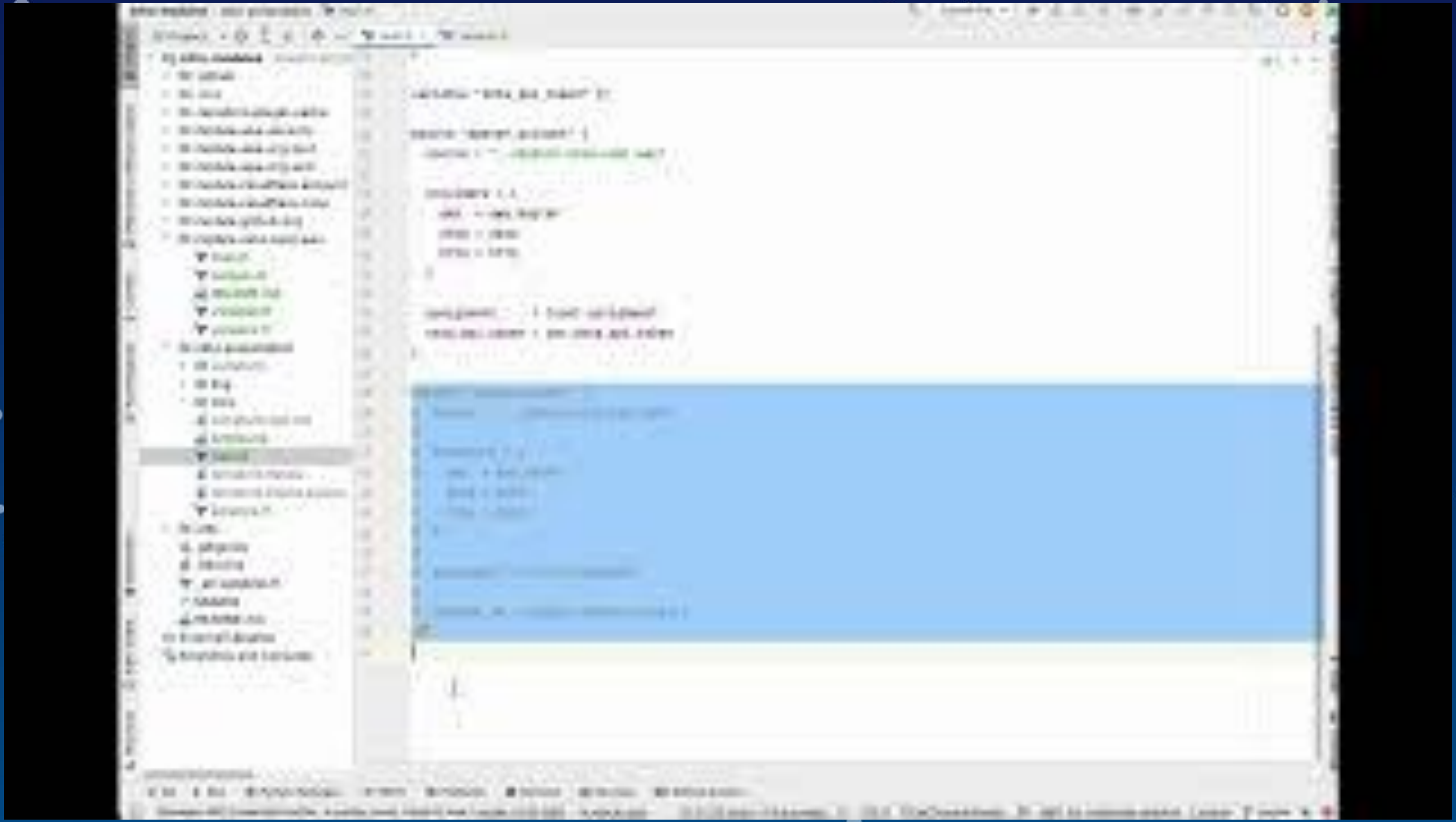
[illegible]

ADDING OTHER ACCOUNTS

For each AWS account IAM user is needed to provision initial AWS resources.

User can be deleted later.

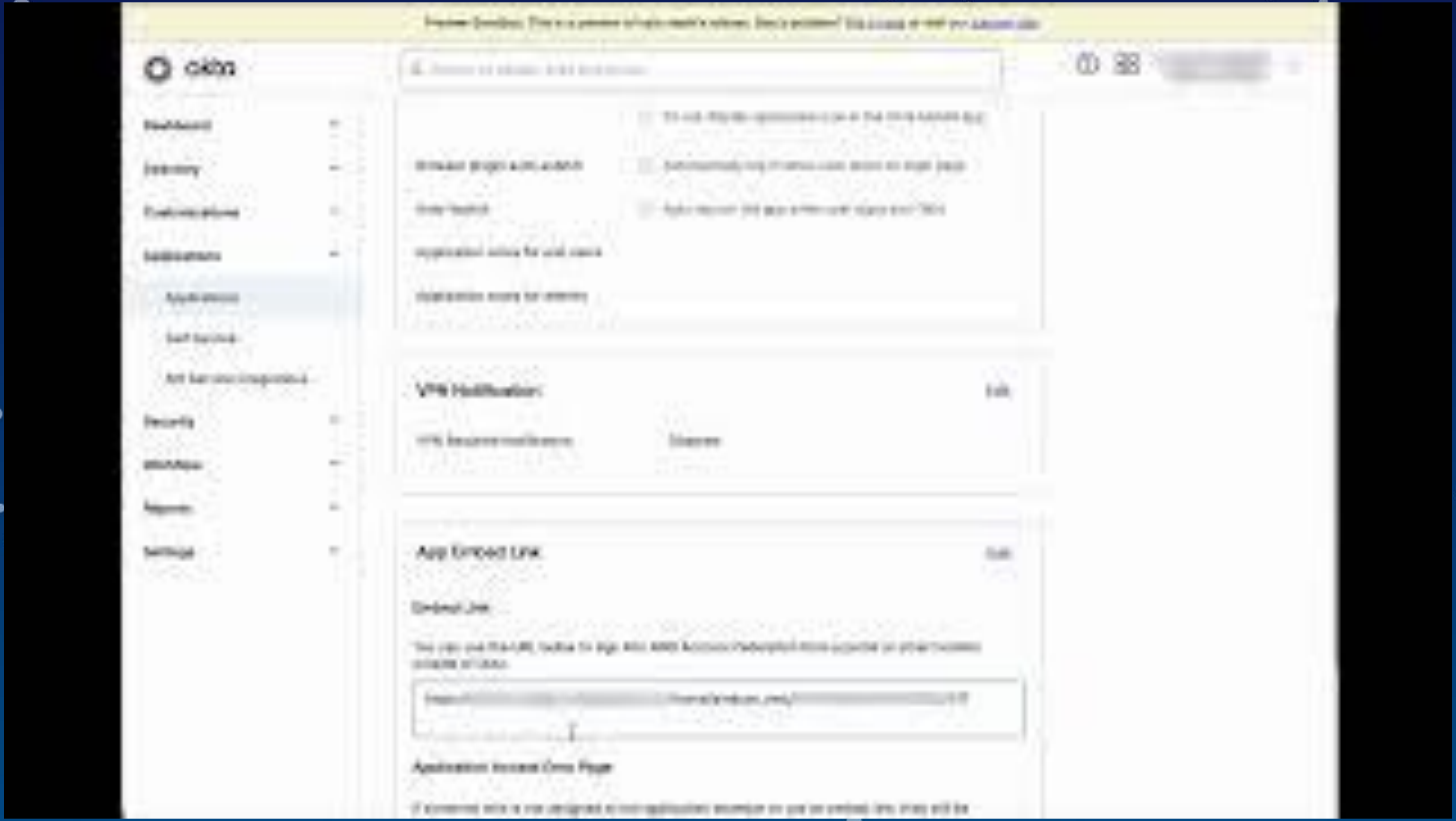
ADD ANOTHER ACCOUNT



SETUP CLI FOR AWS

Setup gimme-aws-creds (presenters’s favorite) to assume IAM roles in terminal. Will require updating profiles for AWS providers in Terraform, but IAM users can be deleted.

CLI FOR TERRAFORM AND TERMINAL



VPN Notification

Edit

App Embed Link

Edit

Embed Link

[redacted]

Application Access Error Page

Summary

Some bonus tips to grow faster

BONUS TIPS

That’s only a beginning, now...

Scale with Terragrunt

With it you don’t need to create IAM users in AWS sub-accounts. It will create IAM role for each one under the hood and use it.

Use auto-provisioning

Assigning to Okta groups can be easily automated, so every new user having specified attribute can gain access to roles and accounts on creation.

Use CI/CD tools

Automating low-level infrastructure and configuration can make any change easier. For example native AWS CodeBuild and CodePipeline.

BRAINLY

THANK YOU

Krzysztof Szyper

MORE INFORMATION

github.com/ChristophShyper/presentation-okta-saml-aws