on: Be sure to plan your installation ahead of time. Your clusters' configurations, as declared in the GKE on-prem uration file, become immutable after you create the clusters. You can't change most aspects of clusters after you hav d them, except for adding or removing nodes. This includes networking and authentication, which must be configure creating clusters.

This page shows how to configure Anthos GKE deployed on-prem to use an OpenID provider for authentication to user clusters. To learn how to use OIDC with AD FS, see Authenticating with OIDC and AD FS (/gke-on-prem/docs/how-to/oidc-adfs).

For an overview of the GKE on-prem authentication flow, see Authentication (/gke-on-prem/docs/concepts/authentication).

GKE on-prem supports OpenID Connect (https://developers.google.com/identity/protocols/OpenIDConnect) (OIDC) as one of the authentication mechanisms for interacting with a user cluster's Kubernetes API server. With OIDC, you can manage access to Kubernetes clusters by using the standard procedures in your organization for creating, enabling, and disabling employee accounts.

There are two ways an employee can use the OIDC authentication flow:

- An employee can use `kubectl` to initiate an OIDC flow. To make this flow automatic, Anthos GKE deployed on-prem provides the Anthos Plugin for Kubectl, a kubectl plugin (https://kubernetes.io/docs/tasks/extend-kubectl/kubectl-plugins/#installing-kubectl-plugins).
- An employee can use Google Cloud Console to initiate an OIDC authentication flow.

In this exercise, you configure both options: `kubectl` and Cloud Console.

This topic assumes you are familiar with OAuth 2.0 (https://oauth.net/2/) and OpenID Connect (https://openid.net/connect/). This topic assumes you are familiar with OpenID scopes (https://openid.net/specs/openid-connect-core-1_0.html#ScopeClaims) and claims (https://openid.net/specs/openid-connect-core-1_0.html#IDToken).

This topic refers to three personas:

- *Organization administrator*: This person chooses an OpenID provider and registers client applications with the provider.
- *Cluster administrator*: This person creates one or more user clusters and creates authentication configuration files for developers who use the clusters.
- *Developer*: This person runs workloads on one or more clusters and uses OIDC to authenticate.

This section is for organization administrators (#personas).

You can use any OpenID provider of your choice. For a list of certified providers, see OpenID Certification (https://openid.net/certification/).

One possibility is to use Active Directory Federated Services (AD FS) (https://docs.microsoft.com/en-us/windows-server/identity/active-directory-federation-services) as your OpenID provider, and use an on-premises instance of Active Directory as your employee database. For more information, see Authenticating with OIDC and AD FS (/gke-on-prem/docs/how-to/oidc-adfs).

This section is for organization administrators (#personas).

As part of establishing a relationship with your OpenID provider, you must specify a redirect URL that the provider can use to return ID tokens to the Anthos Plugin for Kubectl. The Anthos Plugin for Kubectl runs on each developer's local machine and listens on a port of your choice. Choose a port number greater than 1024 that is suitable for this purpose. Then, the redirect URL is:

where ***[PORT]*** is your port number.

When you configure your OpenID provider, specify http://localhost:***[PORT]***/callback as one of your redirect URLs. How you do this depends on your provider.

This section is for organization administrators (#personas).

In addition to having a redirect URL for `kubectl`, you need a redirect URL for Cloud Console. The redirect URL for Cloud Console is:

When you configure your OIDC provider, specify https://console.cloud.google.com/kubernetes/oidc as one of your redirect URLs. How you do this depends on your provider.

This section is for organization administrators (#personas).

Before your developers can use the Anthos Plugin for Kubectl or Cloud Console with your OpenID provider, you need to register those two clients with the OpenID provider. Registration includes these steps:

- Learn the provider's issuer URI. This is where the Anthos Plugin for Kubectl or Cloud Console sends authentication requests.
- Give the provider the redirect URL for the Anthos Plugin for Kubectl.
- Give the provider the redirect URL for Cloud Console. This is https://console.cloud.google.com/kubernetes/oidc.
- Establish a single client ID (https://www.oauth.com/oauth2-servers/client-registration/client-id-secret/). This is the ID that the provider uses to identify both the Anthos Plugin for Kubectl and Cloud Console.
- Establish a single client secret (https://www.oauth.com/oauth2-servers/client-registration/client-id-secret/). The Anthos Plugin for Kubectl and Cloud Console both use this secret to authenticate to the OpenID provider.

- Establish a custom scope that the Anthos Plugin for Kubectl or Cloud Console can use to request the user's security groups.
- Establish a custom claim name that the provider will use to return the user's security groups.

How you perform these steps depends on your OpenID provider. To learn how to perform the registration steps with AD FS, see Authenticating with OIDC and AD FS (/gke-on-prem/docs/how-to/oidc-adfs).

This section is for cluster administrators (#personas).

Before you create a user cluster, you generate a GKE on-prem configuration file using `gkectl create-config`. The configuration includes the following `oidc` specification. You populate `oidc` with values specific to your provider:

- `issuerurl`: Required. URL of your OpenID provider, such as `https://example.com/adfs`. Client applications, like the Anthos Plugin for Kubectl and Cloud Console, send authorization requests to this URL. The Kubernetes API server uses this URL to discover public keys for verifying tokens. Must use HTTPS.
- `kubectlredirecturl:` Required. The redirect URL configured previously for the Anthos Plugin for Kubectl.

- `clientid`: Required. ID for the client application that makes authentication requests to the OpenID provider. Both the Anthos Plugin for Kubectl and Cloud Console use this ID.
- `clientsecret`: Optional. Secret for the client application. Both the Anthos Plugin for Kubectl and Cloud Console use this secret.
- `username`: Optional. JWT claim to use as the username. The default is `sub`, which is expected to be a unique identifier of the end user. You can choose other claims, such as `email` or `name`, depending on the OpenID provider. However, claims other than `email` are prefixed with the issuer URL to prevent naming clashes with other plugins.
- `usernameprefix`: Optional. Prefix prepended to username claims to prevent clashes with existing names. If you do not provide this field, and `username` is a value other than `email`, the prefix defaults to `issuerurl#`. You can use the value `-` to disable all prefixing.
- `group`: Optional. JWT claim that the provider will use to return your security groups.
- `groupprefix`: Optional. Prefix prepended to group claims to prevent clashes with existing names. For example, given a group `foobar` and a prefix `gid-`, `gid-foobar`. By default, this value is empty, and there is no prefix.
- `scopes`: Optional. Additional scopes to send to the OpenID provider as a comma-delimited list.
- `extraparams`: Optional. Additional key-value parameters to send to the OpenID provider as a comma-delimited list.
    - For a list of authentication parameters, see Authentication URI parameters (https://developers.google.com/identity/protocols/OpenIDConnect#authenticationuriparameters)
    - For a list of Microsoft Azure's authentication parameters, see Send the sign-in request (https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-protocols-oidc#send-the-sign-in-request)
      .
    - If you are authorizing a group (#example-group), pass in `resource=token-groups-claim`.
    - If your authorization server prompts for consent (https://developers.google.com/identity/protocols/OpenIDConnect#prompt), pass in `prompt=consent`.
- `usehttpproxy`: Optional. Specifies whether to deploy a reverse proxy in the cluster to allow Connect Agent access to the on-premises OIDC provider for authenticating users. Value must be a string: `"true"` or `"false"`.
- `capath`: Optional. Path to the certificate for the certificate authority (CA) that issued your identity provider's web certificate. This value might not be necessary. For example, if your identity

provider's certificate was issued by a well-known public CA, then you would not need to provide a value here.

Many providers encode user-identifying properties, such as email and user IDs, in a token. However, these properties have implicit risks for authentication policies:

- User IDs can make policies difficult to read and audit.
- Emails can create both an availability risk (if a user changes their primary email) and potentially a security risk (if an email can be re-assigned).

Therefore, it's a best practice to use group policies, as a group ID can be both persistent and easier to audit.

Suppose your provider creates identity tokens that include the following fields:

Given this token format, you'd populate your configuration file's `oidc` specification like so:

After you've created the user cluster, you could then use Kubernetes role-based access control (RBAC) to grant privileged access to the authenticated users. For example, you could create a ClusterRole that grants its users read-only access to the cluster's Secrets, and create a ClusterRoleBinding resource to bind the role to the authenticated group:

This section is for cluster administrators (#personas).

After you create a user cluster, create an authentication configuration file for your cluster:

where ***[USER_CLUSTER_KUBECONFIG]*** is the path of your user cluster's kubeconfig file. When you created your user cluster, `gkectl create cluster` generated a kubeconfig file for the cluster.

The preceding command creates a file named `kubectl-anthos-config.yaml` in the current directory.

This section is for cluster administrators (#personas).

You can store the authentication configuration for several clusters in a single file. Then you can distribute that one file to the developers who need access to all of those clusters.

Start with an existing authentication configuration file. Then use the following command to merge that file with the configuration for an additional cluster:

where

- ***[USER_CLUSTER_KUBECONFIG]*** is the kubeconfig file of the additional cluster.
- ***[IN_AUTH_CONFIG_FILE]*** is the path of the existing authentication configuration file.
- ***[OUT_AUTH_CONFIG_FILE]*** is the path where you want to save the file that holds the merged configuration. This can be the same as ***[IN_AUTH_CONFIG_FILE]***.

In a single authentication configuration file, the individual cluster names must be unique.

This section is for cluster administrators (#personas).

The authentication configuration file that you distribute to your developers must be named`kubectl-anthos-config.yaml`. You can distribute the authentication configuration file in a number of ways:

- Manually give the file to each developer.
- Host the file at a URL, so developers can download it.
- Push the file to each developer's machine.

Regardless of your distribution method, for a default configuration, the authentication configuration file must be placed at this location on each developer's machine:

If you don't want to use the default configuration, you can create a custom configuration (#custom_configuration).

This section is for developers (#personas).

The authentication configuration file contains the details of all clusters you can authenticate to. Do not modify the contents of this file.

If your cluster administrator has provided you with an authentication configuration file, place the file in the correct directory. If your cluster administer has already placed the configuration on your machine, you can skip this section.

Copy your authentication configuration file to its default location:

lf you do not want to set up a default configuration, you can manually pass the path of the authentication configura
:he plugin commands by using the **`login-config`** flag. For example, see Using the Anthos Plugin for Kubectl to
nticate (#plugin_to_authenticate).

This section is for cluster administrators (#personas).

The Anthos Plugin for Kubectl is included in your admin workstation at:

You can distribute the plugin to your developers, or you can have them download the plugin directly.

This section is for cluster administrators and developers (#personas).

Each developer needs to have the Anthos Plugin for Kubectl on their own machine. Developers can download the plugin individually, or the cluster administrator can download the plugin and then distribute it to the developers.

Note for developers: Ask your cluster administrator what version of the plugin you should use.

Download the Anthos Plugin for Kubectl:

This section is for developers (#personas).

Your cluster administrator might provide you with the Anthos Plugin for Kubectl. Or your cluster administrator might tell you to download the plugin (#download_plugin) yourself.

To install the Anthos Plugin for Kubectl, move the executable file to any location on your PATH. For Linux and macOS, the executable file must be named `kubectl-anthos`. For Windows, it must be named `kubectl-anthos.exe`. To learn more, see the Installing kubectl plugins (https://kubernetes.io/docs/tasks/extend-kubectl/kubectl-plugins/#installing-kubectl-plugins).

Verify that the Anthos Plugin for Kubectl is installed:

Some Windows users might have to run the command as **`kubectl-anthos.exe version`** instead of **`kubectl ant`** **`on`**.

Extending kubectl with plugins is supported in kubectl v1.12 and later.

This section is for developers (#personas).

If your authentication configuration file is at the default path, enter this command to list the clusters that you can authenticate to:

If your authentication configuration file is not at the default path, enter this command to list clusters:

where ***[AUTH_CONFIG_FILE]*** is the path to your authentication configuration file.

This section is for developers (#personas).

Log in to a user cluster:

where:

- ***[CLUSTER_NAME]*** is the name of your user cluster. This name must be selected from the list provided by the `kubectl anthos listclusters` command.
- ***[USER_NAME]*** an optional parameter that specifies the username for the credentials stored in the kubeconfig file. The default value is `[CLUSTER_NAME]-anthos-default-user`.
- ***[AUTH_CONFIG_FILE]*** is the path of your authentication configuration file. If your authentication configuration file is in the default (#placing_auth_config) location, you can omit this parameter.
- ***[USER_CLUSTER_KUBECONFIG]*** is the path of your user cluster's kubeconfig file. If a kubeconfig file does not exist, the command generates a new kubeconfig file from the authentication configuration file, and adds the cluster details and tokens to the kubeconfig file.

`kubectl anthos login` launches a browser where you can enter your credentials.

The kubeconfig file provided now contains an ID token that `kubectl` can use to authenticate to the Kubernetes API server on the user cluster.

Some Windows users might have to run the command as `kubectl-anthos.exe login` instead of `kubectl antho`
.

The `kubectl anthos login` command has an optional `--dry-run` flag. If you include the `--dry-run` flag, the command prints the `kubectl` commands that would add tokens to the kubeconfig file, but does not actually save the tokens in the kubeconfig file.

Verify that authentication was successful by entering a `kubectl` command. For example:

This section is for developers (#personas).

1. Verify that your cluster is configured for OIDC (#oidc_spec).
2. Verify that your cluster has been registered with Google Cloud (/anthos/multicluster-management/connect/registering-a-cluster), either automatically during cluster creation or manually.
3. Visit the **Kubernetes clusters** page in Cloud Console.

   Visit the Kubernetes clusters page (https://console.cloud.google.com/kubernetes/list)
4. In the list of clusters, locate your GKE On-Prem cluster, and click **Login**.
5. Select **Authenticate with the Identity Provider configured for the cluster**, and click **LOGIN**.

   You will be redirected to your identity provider, where you might need to log in or consent to Cloud Console accessing your account. Then you will be redirected back to the **Kubernetes clusters** page in Cloud Console.

By default, the Anthos Plugin for Kubectl expects the authentication configuration file to be at a certain location (/gke-on-prem/docs/how-to/oidc#distributing_the_authentication_configuration_file). If you do not want to put the authentication configuration file at the default location, you can manually pass the path of the authentication configuration file to the plugin commands by using the `--login-config` flag. For example, see Using the Anthos Plugin for Kubectl to authenticate (#plugin_to_authenticate).

If Cloud Console cannot read the OIDC configuration from your cluster, the **LOGIN** button will be disabled.

If your identity provider configuration is invalid, you will see an error screen from your identity provider after you click **LOGIN**. Follow the provider-specific instructions to correctly configure the provider or your cluster.

If you complete the authentication flow, but still don't see the details of the cluster, make sure you granted the correct RBAC permissions to the account that you used with OIDC. Note that this might be a different account from the one you use to access Cloud Console.

You might hit this error if the authorization server prompts for consent, but the required authentication parameter wasn't provided. Provide the `prompt=consent` parameter to GKE on-prem configuration file's `oidc: extraparams` field, and regenerate the client authentication file with the `--extra-params prompt=consent` flag.