# Accessing GitHub in Substrate



Document Level Classification

200



# Introduction

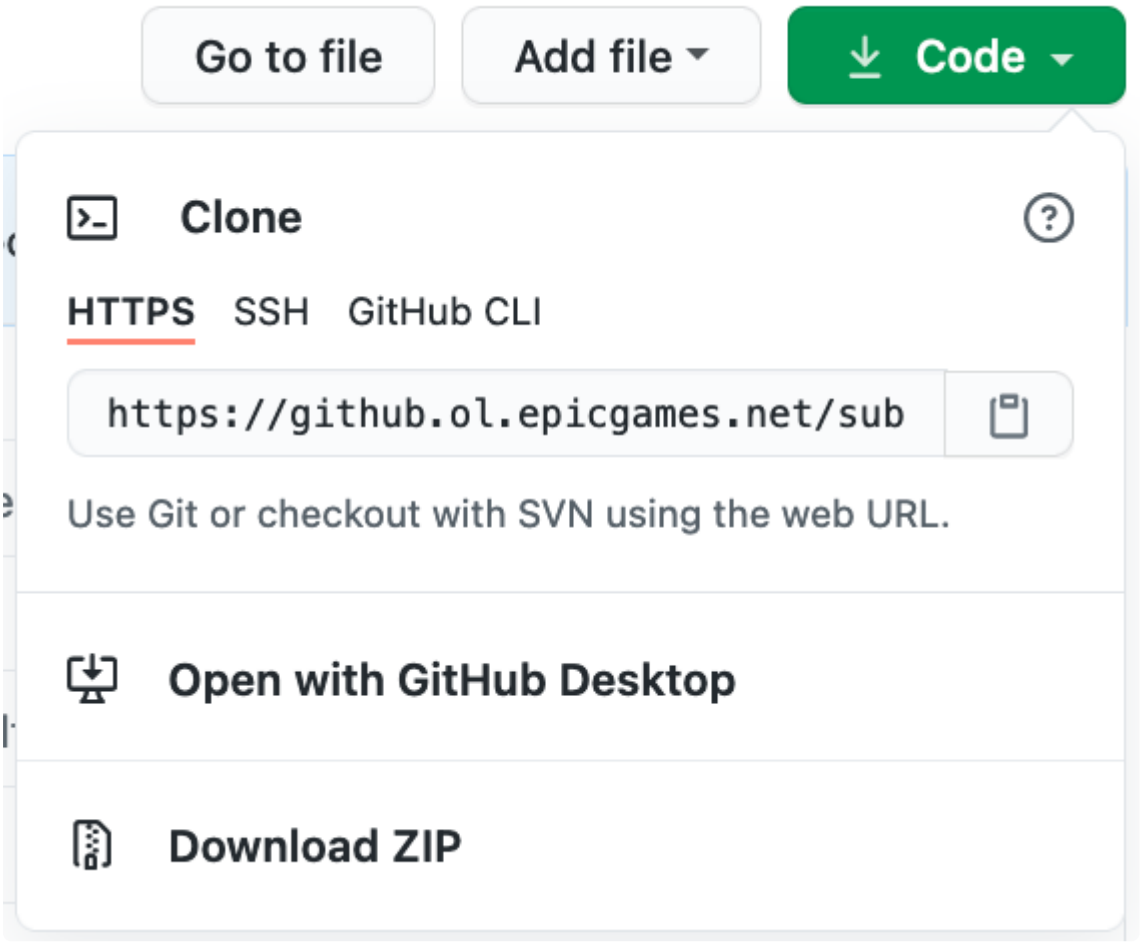
When interacting with Substrate we assume you are familiar with Git and GitHub. Our workflows assume your code and configuration are hosted internally https://github.ol.epicgames.com. Here are a handful of tips for working with GitHub in Substrate.

# Cloning Repositories

We recommend using **HTTPS** clones and references to GitHub. HTTPS is a layer 7 protocol and supports a broader range of network configurations. SSH may work most of the time, but HTTPS will work everywhere in Substrate.

Pay attention to the protocol when including references to Git repositories in Terraform projects, Git submodules, and package management tools like `go mod` and `npm`. These also encode the Git address and can use SSH or HTTPS addresses. Prefer HTTPS in these places also.

When you select links to clone from the GitHub UI, you can select HTTPS from this dropdown menu.



# URL rewriting

The git CLI supports automatic rewriting of URLs, even from one protocol to another.  This can be utilized to ensure clones and updates in a Substrate environment are done over HTTPS, even if SSH is being requested.

```
git config --global url."https://github.ol.epicgames.net/".insteadof "g
```

creates the following entry in .gitconfig:

```
[url "https://github.ol.epicgames.net/"]
insteadof = git+ssh://git@github.ol.epicgames.net/
```

# Git credential.helper

When you first try to clone over HTTPS, you'll notice that GitHub prompts for your username and password. Before you type your Okta creds into the prompt, let's take a moment to setup credential caching. Each OS has a different way of doing this, but the result is that your HTTPS credentials are saved in a secure store in the OS and you only need to provide them once.

Read more in GitHub's documentation: https://docs.github.com/en/get-started/getting-started-with-git/caching-your-github-credentials-in-git

Saved Git credentials use a Personal Access Token. You can create one yourself or see your current tokens in the GitHub UI. Git requires **repo** access permissions to clone and push.

## Linux

In linux the tool is typically included in your Git installation but may need to be compiled. For example, on Ubuntu 18.04:

```
sudo apt install -y make gcc libglib2.0-dev libsecret-1-dev --no-instal
cd /usr/share/doc/git/contrib/credential/libsecret
make
sudo mv git-credential-libsecret `git --exec-path`
git config --global --add credential.helper libsecret
```

On Gnome desktop environments your credential will be stored in the `gnome-keyring`. You can use `seahorse` to manage saved credentials.

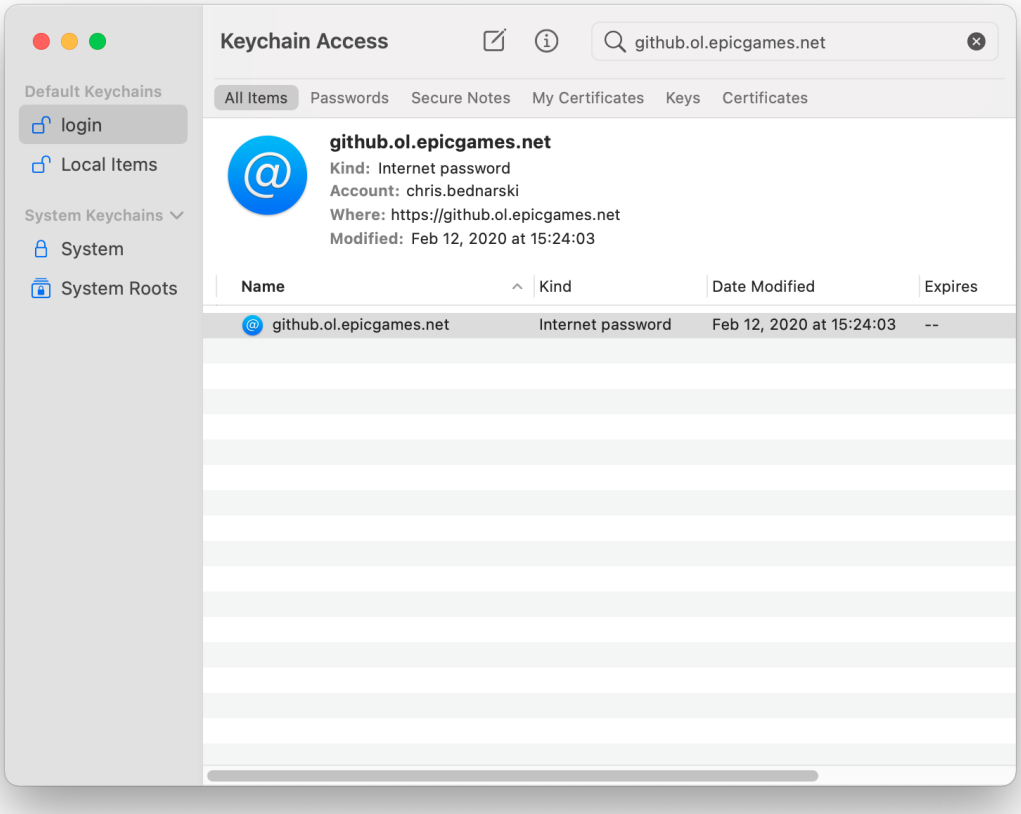
## MacOS

XCode includes the credential helper but it is not configured by default. You can do so by adding it to your Git configuration:

```
git config --global --add credential.helper osxkeychain
```

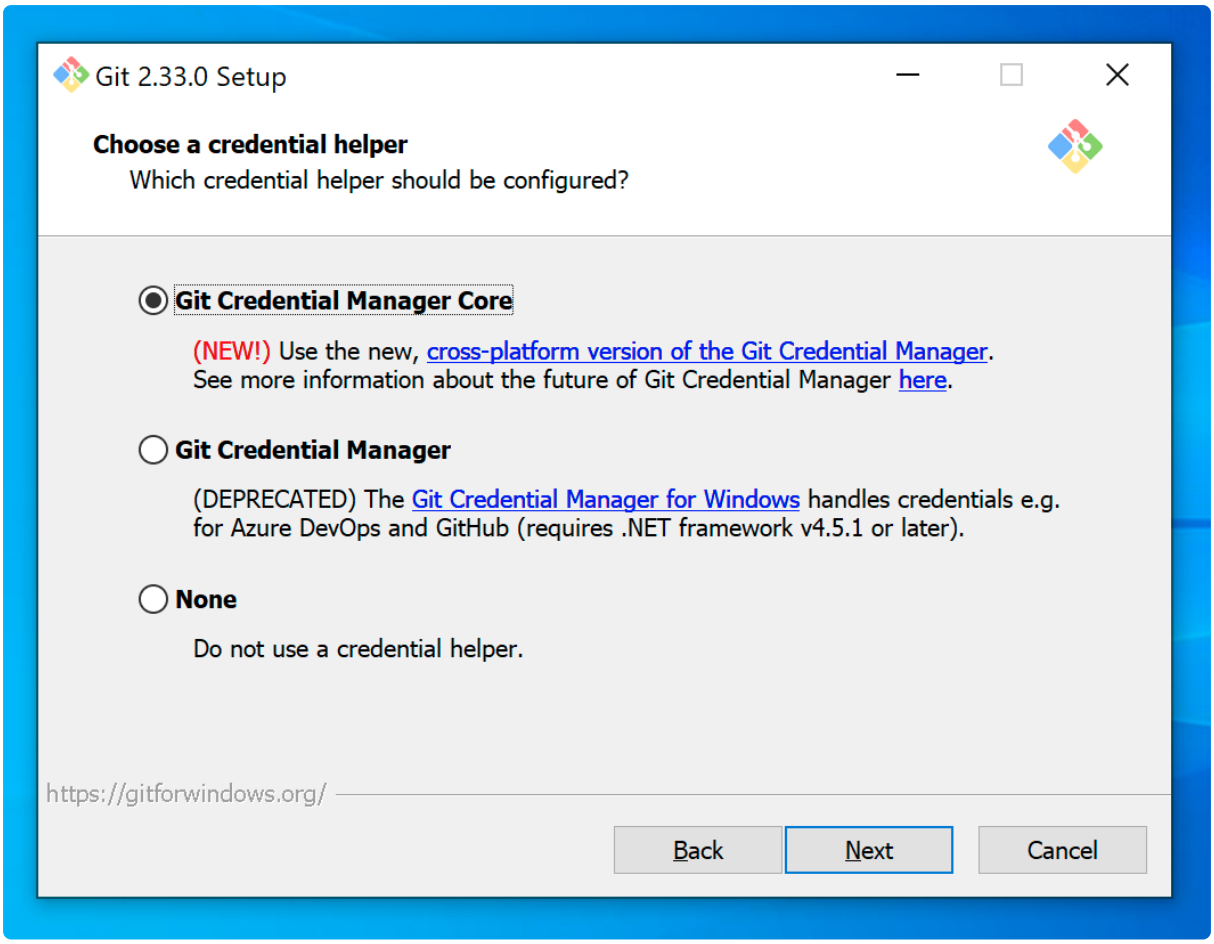
After your first successful git clone your credentials will be saved in the system Keychain.

You can use Keychain Access to manage saved credentials.

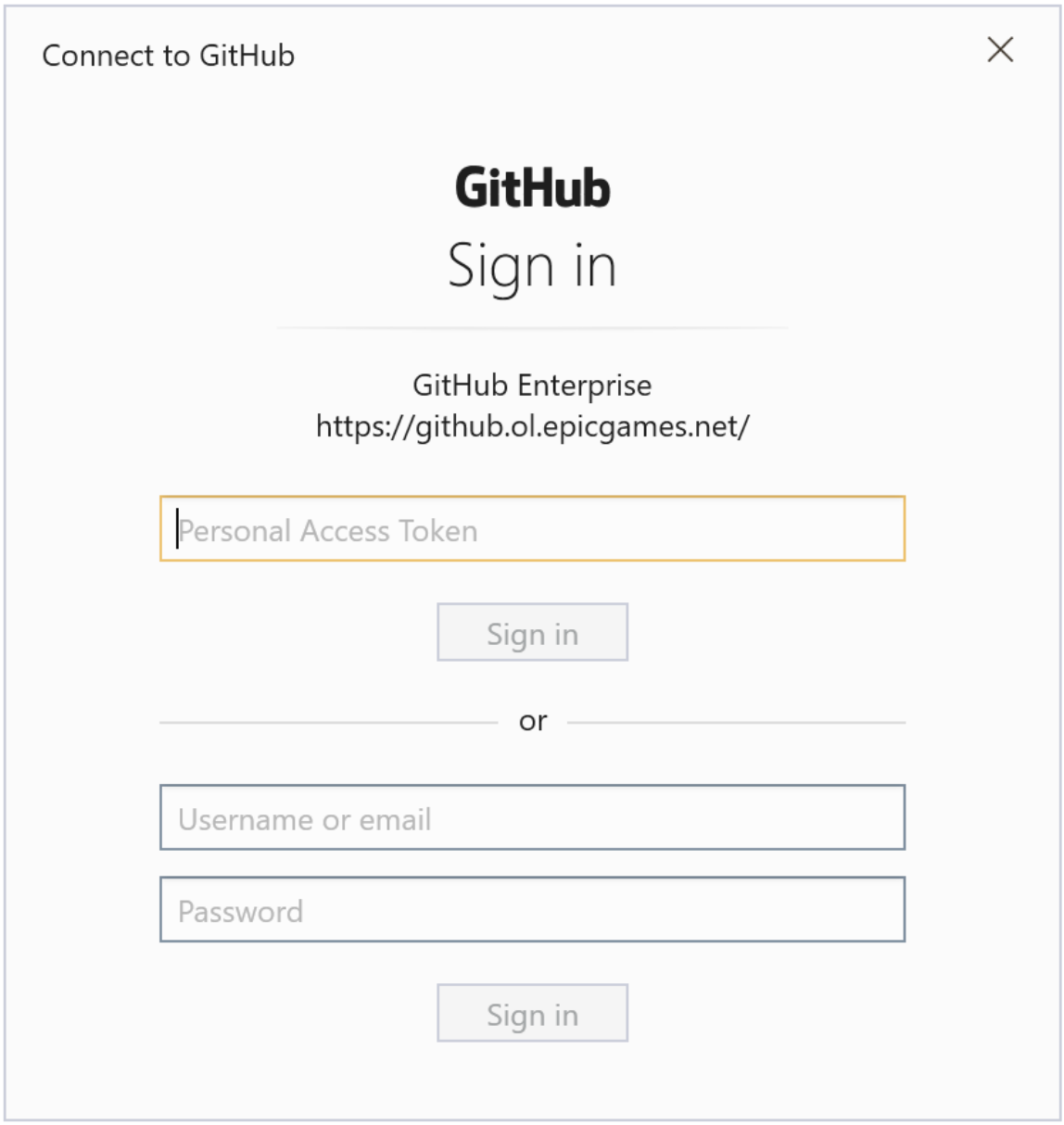



## Windows

The Git for Windows installer includes a credential manager called Git Credential Manager Core. Select this option during installation or upgrade and it will automatically cache your Git HTTPS credentials in a secure store in Windows.

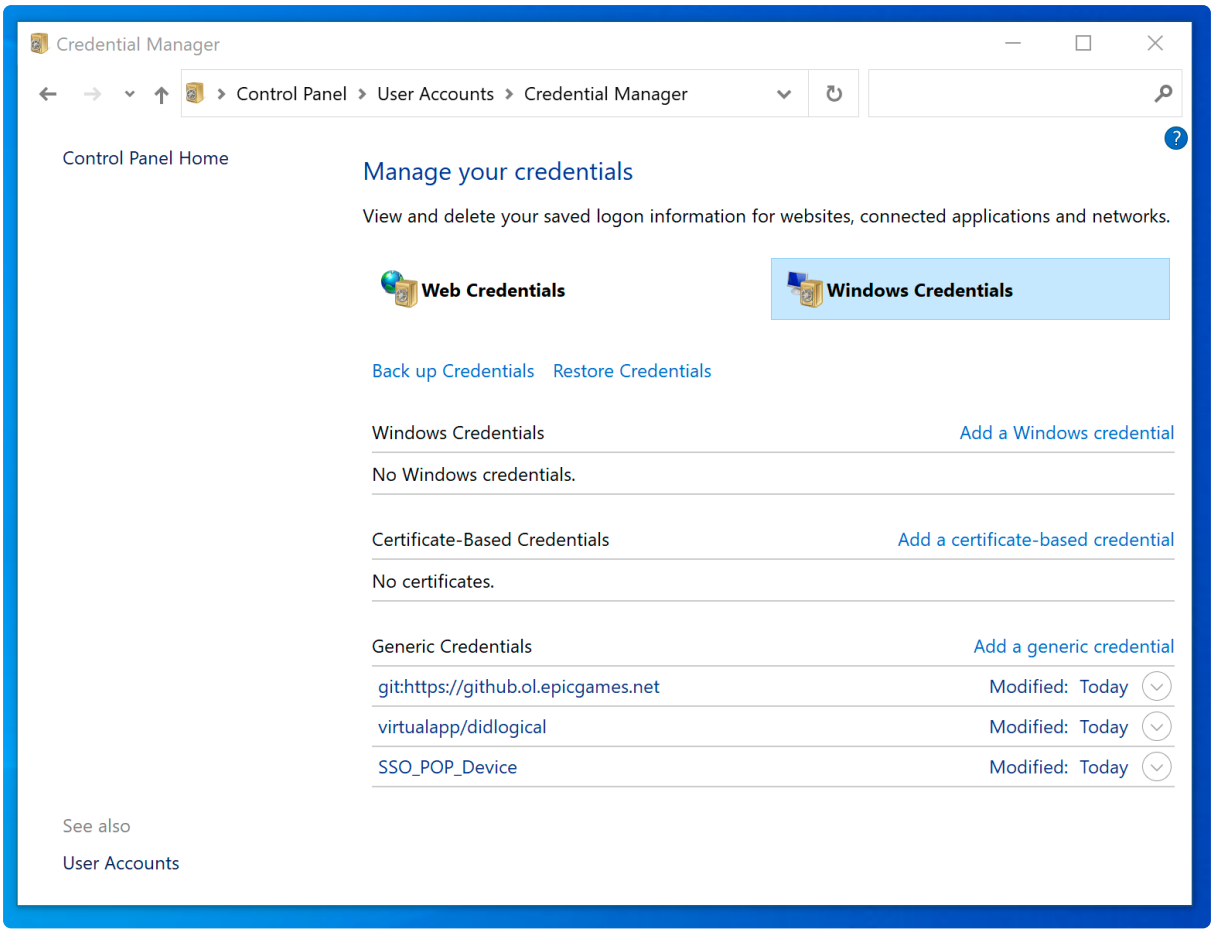



When you first clone a repo in Git you will see a pop-up prompting for your username and password:




Enter your personal access token in the dialog.

Afterward you can manage your saved credentials in the Windows Control Panel under Credentials Manager:



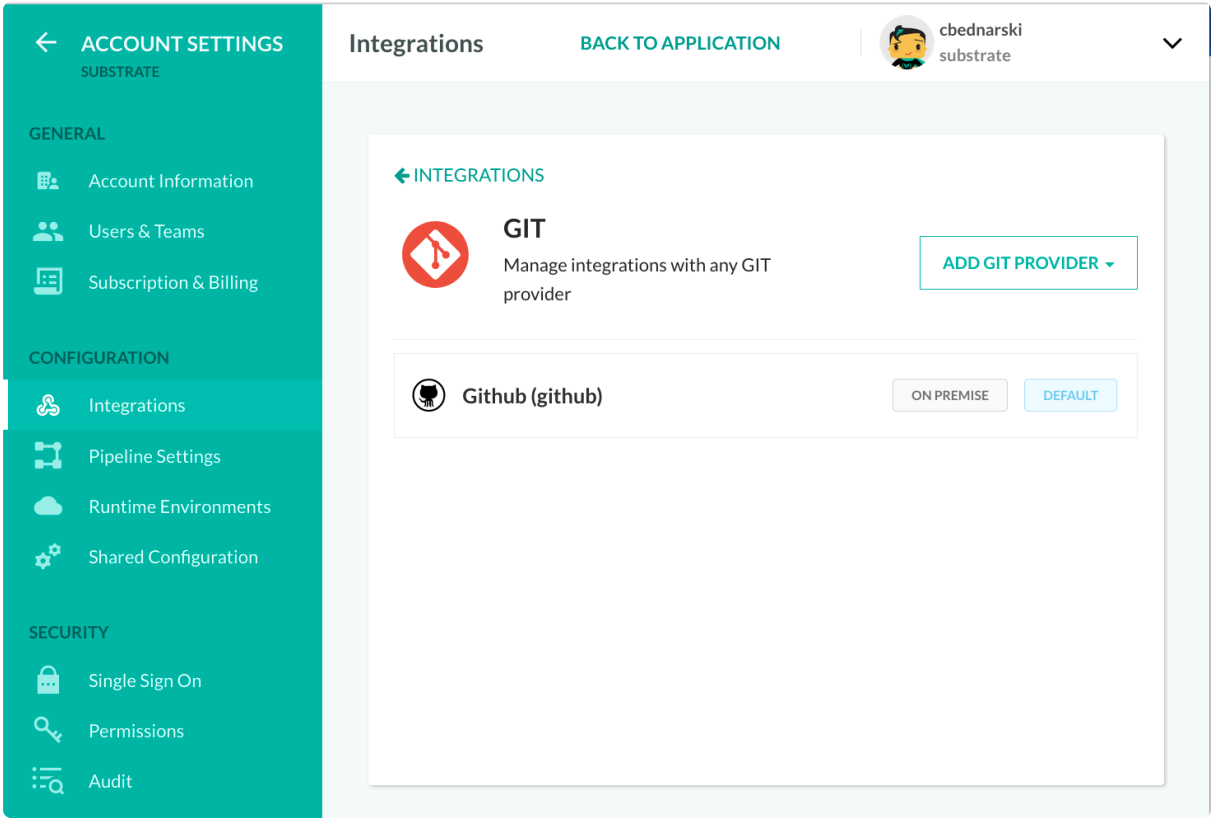

# How do I fetch code from GitHub in CodeFresh?

## Cloning Repositories

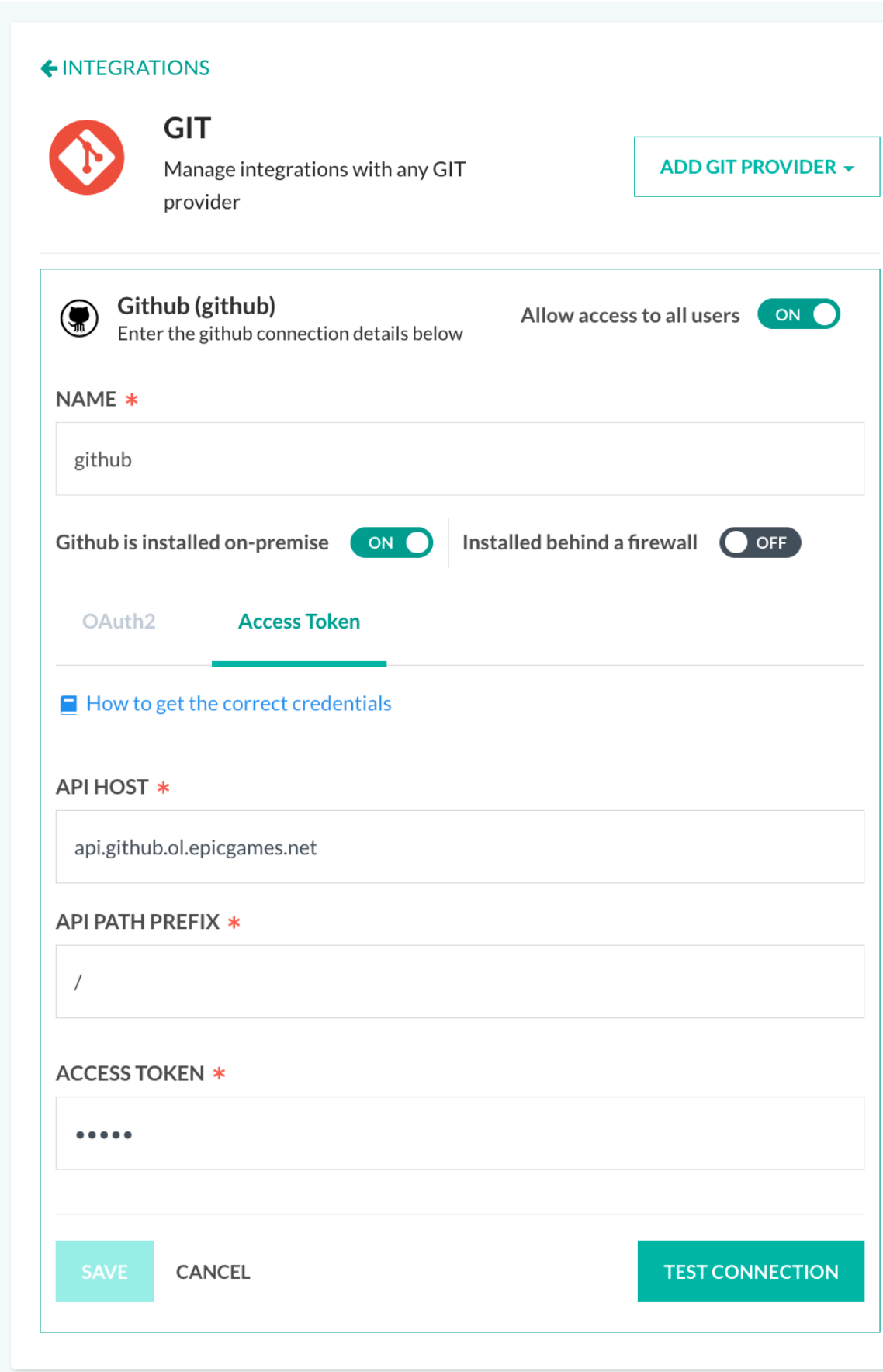
When you first login to a new CodeFresh account you may need to configure GitHub access. Create a new personal access token for CodeFresh. CodeFresh requires **repo**, **admin:org**, and **admin:repo_hook** permissions.

When you add configuration using your account, other users who access GitHub will do so using your credentials and access. You can request a service account from IT if you want to create a separate user or grant more granular permissions.

Once you have the access token you will configure a  new Git integration under under [Account Settings → Integrations → Git](#).




Copy the settings you see below and include your personal access token.


INTEGRATIONS
GIT
Manage integrations with any GIT provider
ADD GIT PROVIDER
Github (github)
Enter the github connection details below
Allow access to all users
ON
NAME *
github
Github is installed on-premise
ON
Installed behind a firewall
OFF
OAuth2
Access Token
How to get the correct credentials
API HOST *
api.github.ol.epicgames.net
API PATH PREFIX *
/
ACCESS TOKEN *
•••••
SAVE
CANCEL
TEST CONNECTION

Afterward, your builds will be able to reference GitHub using the provider name you specified above (in this case `github`). For example:

```
steps:
  clone:
    title: "Cloning repository"
    type: "git-clone"
    git: github  <--- Matches the NAME field in the integration setting
    repo: "https://github.ol.epicgames.net/substrate/starmap/"
    revision: "${{CF_REVISION}}"
```

If you use other names, you can configure multiple ways for CodeFresh to access GitHub, each with different permissions for different projects.

## Build Dependencies

It's common for package management tools like `npm` and `go get` to resolve dependencies at build time. However, this practice should be avoided. Instead, resolve your dependencies during development and check them into your project as vendored code. This accomplishes several things:

1. Dependencies always match your code because they are checked into source control
2. Dependencies that are checked into your code can't spontaneously disappear from the internet and break your build
3. Builds are faster because dependencies are downloaded from the internet once
4. Changes to dependencies can be reviewed alongside changes to other code
5. Dependencies are downloaded in CI along with your code, so they do not need to be fetched separately over the network

## Other Access

If you want to use the GitHub token configured in CodeFresh for some other calls to the GitHub API (for example, to push tags or make releases), you can retrieve via `codefresh.yml`. For example:

```
version: "1.0"
steps:
  getGitToken:
    title: Reading GitHub token
    image: codefresh/cli
    commands:
      - cf_export GITHUB_TOKEN=$(codefresh get context github --decrypt
```

This becomes available for subsequent steps and commands via the `GITHUB_TOKEN` environment variable.

# Accessing GitHub from Kubernetes or AWS

If you want to access GitHub directly from Kubernetes or AWS you will first need to make sure that `github.ol.epicgames.net` is resolvable in your VPC.

Here is an example VPC configuration which includes the github.ol.epicgames.net name: https://github.ol.epicgames.net/substrate/infrastructure/blob/master/accounts/dead-dev/vpc.tf#L33-L38

You can ask for help in #cloud-support-ext in Slack to configure this in your VPC.

---