Feature Flags

Lab Manual

Contents

Lab : Feature Flags

Introduction

In this lab, you will work with LaunchDarkly to implement feature flags, exploring their available functionality.

Objectives

After completing this lab, you will be able to:

- Use Feature Flags effectively

Prerequisites

None

Estimated Time to Complete This Lab

30 minutes

For More Information

Microsoft and LaunchDarkly: https://launchdarkly.com/microsoft/

LaunchDarkly: https://launchdarkly.com/

Exercise 1: Create a feature flag

Introduction

Feature flags are an important DevOps technique that makes the process of continuous delivery a lot easier. They help with the following aspects of the release process:

- Nearly eliminate integration (merge) conflicts.
- Toggle features to hide, disable, or enable features at run-time.
- Revert a change deployed to production without rolling back your release.
- Present users with variants of a feature, to determine which one performs better.

Objectives

After completing this lab, you will be able to:

- Understand and control feature flags to turn features on/off

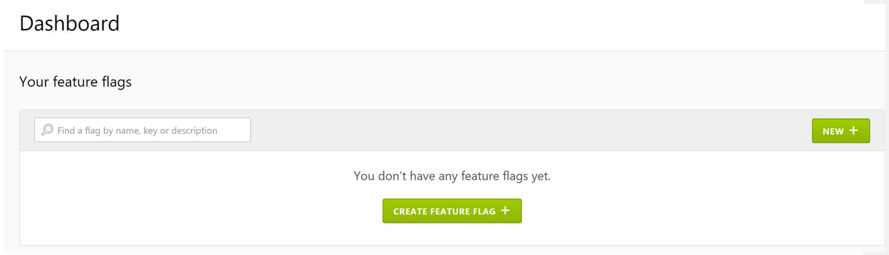
Prerequisites

None

Tasks

Implementing feature flags can either be achieved by building your own infrastructure or by leveraging one of the many products in the marketplace. In this hands-on lab, we have chosen the latter. We will use a product called LaunchDarkly. **Our choice is not a recommendation of the product**. You should evaluate the solution that best meet your needs.

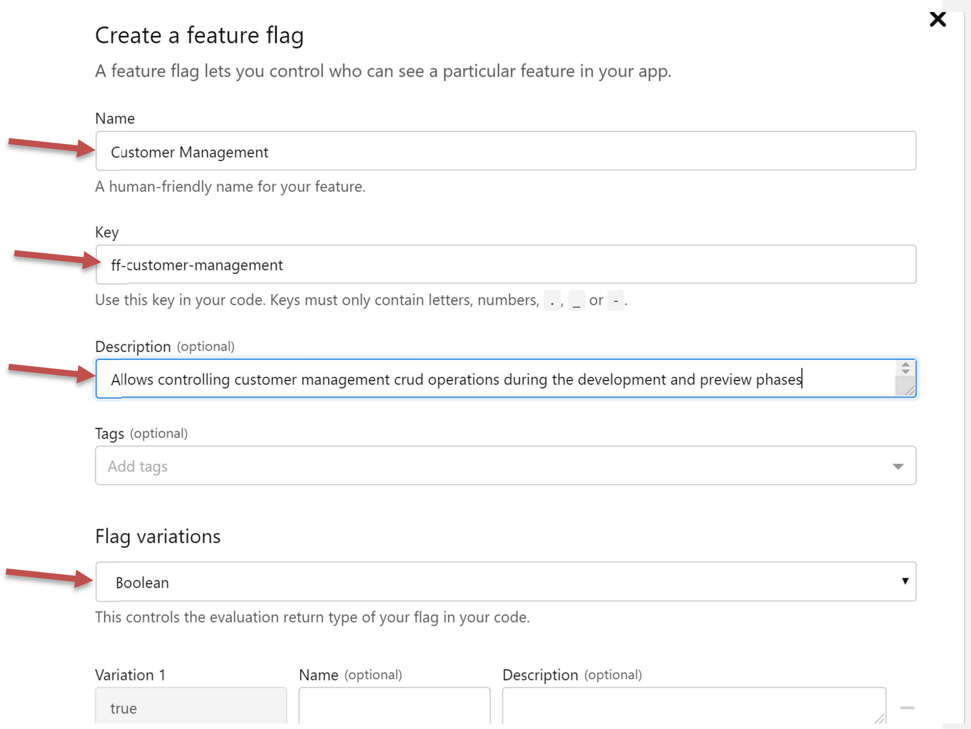
- Go to this web site and sign up for a **free trial version**: https://launchdarkly.com/start/index.html#signup
- Open a shell window (Bash, Windows Command Line or PowerShell). You can use the integrated command line from within VS Code.
- Go to the solution root folder of your "ModernUISolution", and then to the "ModernUIApp" folder: "cd ModernAppUISolution" (Make sure this name matches the name you chose for the UI solution folder)
- From the "LaunchDarkly" portal add a new entry for the feature flag:


Dashboard

Your feature flags

You don't have any feature flags yet.

- Click on "Create feature flag", then enter the following information:



Create a feature flag

A feature flag lets you control who can see a particular feature in your app.

Name: Customer Management

A human-friendly name for your feature.

Key: ff-customer-management

Use this key in your code. Keys must only contain letters, numbers, . , _ or - .

Description (optional): Allows controlling customer management crud operations during the development and preview phases

Tags (optional): Add tags

Flag variations: Boolean

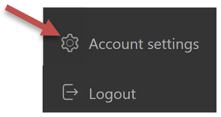
This controls the evaluation return type of your flag in your code.

Variation 1	Name (optional)	Description (optional)
true		

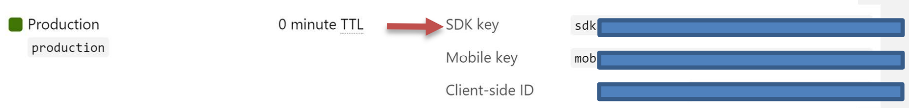
Click “Save Flag”.

Note: Make sure you take note of the feature flag key you entered in this screen, you will refer to it in the rest of this lab.

- From the “Launchdarkly” dashboard, go to the account settings.

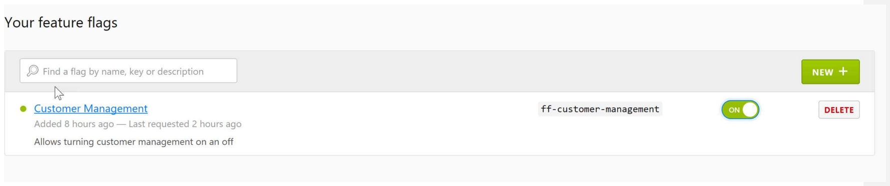


- Copy the production SDK key for the “Production” environment. You will need it to integrate the code with the feature flag:



🟩 Production `production`	0 minute TTL	SDK key	sdk[redacted]
		Mobile key	mob[redacted]
		Client-side ID	[redacted]

- Go to the feature flags dashboard and click on the feature flag you created.


Your feature flags

Find a flag by name, key or description

NEW +

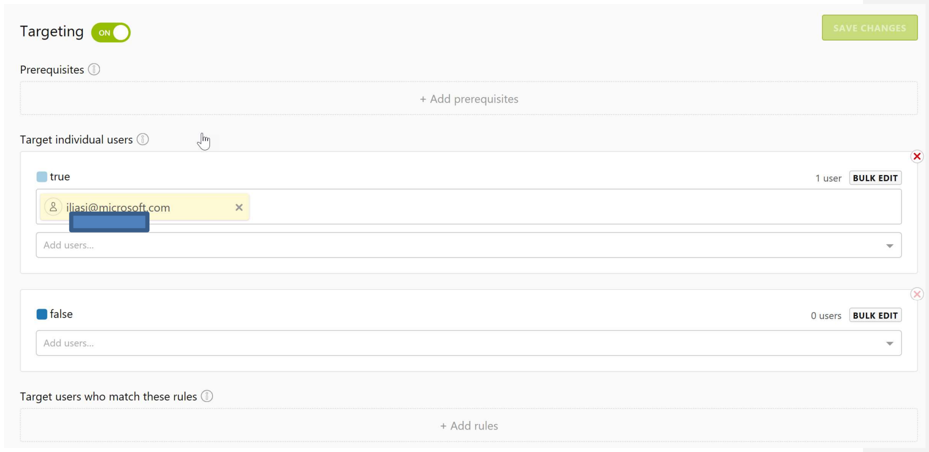
● Customer Management
Added 8 hours ago — Last requested 2 hours ago
Allows turning customer management on an off

`ff-customer-management`

ON

DELETE

- Click on the "Customer Management" feature flag:
- Enable targeting, and add a user in the true section (you can enter your own email address for instance):


Targeting ON

SAVE CHANGES

Prerequisites

+ Add prerequisites

Target individual users

true — 1 user — BULK EDIT

iljasi@microsoft.com [redacted]

Add users...

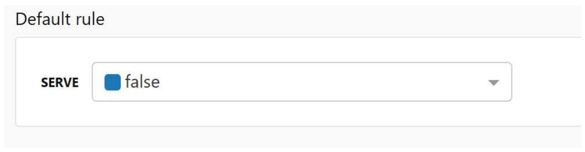
false — 0 users — BULK EDIT

Add users...

Target users who match these rules

+ Add rules

- Make sure the default rule is false:


Default rule

SERVE false

- Save the changes you've made.

Exercise 2: Use a feature flag in your application

Objectives

After completing this lab, you will be able to:

- Integrate feature flags with your code.

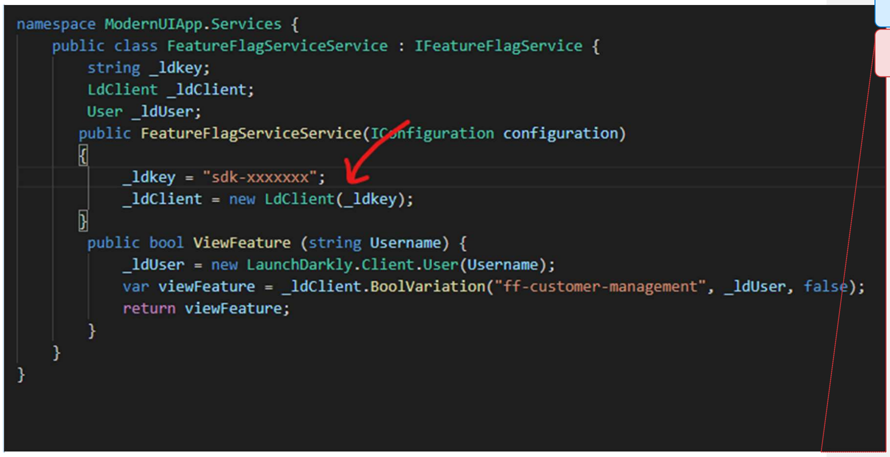
Prerequisites

None

- Replace the content of your "ModernUIAppSolution" folder with the content of the "..\Exercice 2\ModernUIAppSolution" folder.
  We are now ready to test the full application interaction.
- Open a PowerShell window and go to the root folder containing both your "ModernApiAppSolution" and your "ModernUIAppSolution".
- Use the following command to open Visual Studio Code:
  Code .
- Open the "FeatureFlagService" and replace the "LaunchDarklyKey" environment variable with the key you have copied in the previous step.

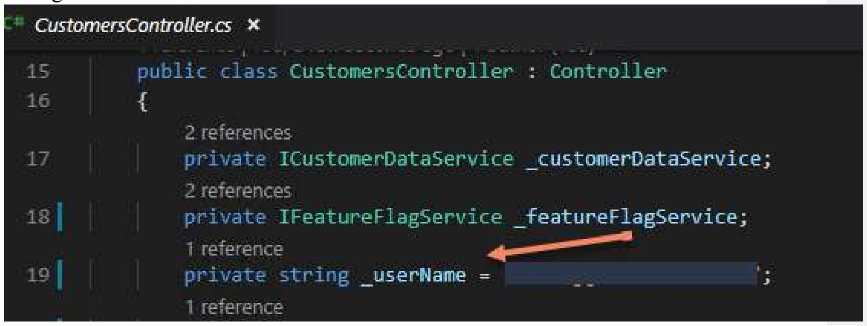
Commented [JB1]: I had issues with this and had to make certain the ASPNETCORE_ENVIRONMENT was the same as the environment defined in the Account Settings in Launch Darkly.

Commented [IJ2R1]: Indeed.
Need two keys for Dev and Prod.



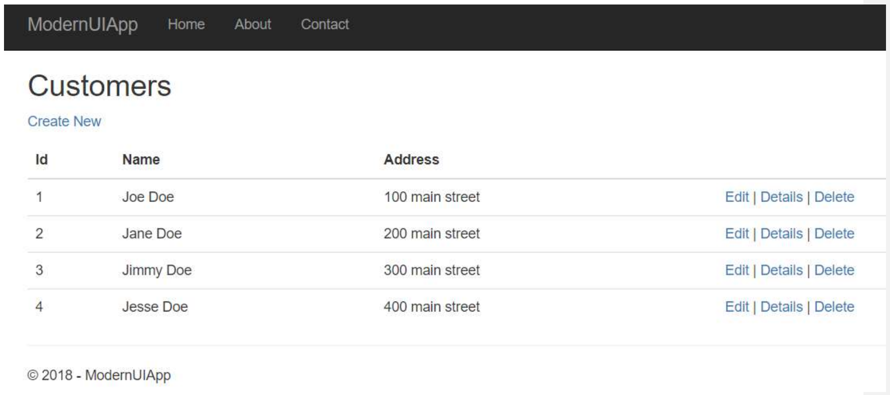
```
namespace ModernUIApp.Services {
    public class FeatureFlagServiceService : IFeatureFlagService {
        string _ldkey;
        LdClient _ldClient;
        User _ldUser;
       public FeatureFlagServiceService(IConfiguration configuration)
       {
            _ldkey = "sdk-xxxxxxx";
            _ldClient = new LdClient(_ldkey);
       }
        public bool ViewFeature (string Username) {
            _ldUser = new LaunchDarkly.Client.User(Username);
            var viewFeature = _ldClient.BoolVariation("ff-customer-management", _ldUser, false);
            return viewFeature;
        }
    }
}
```

- Change the username in the "CustomersController.cs"



```
public class CustomersController : Controller
{
    2 references
    private ICustomerDataService _customerDataService;
    2 references
    private IFeatureFlagService _featureFlagService;
    1 reference
    private string _userName = [redacted];
    1 reference
```

- Run the project
- Once the application runs go to the page: http://localhost:9090/Customers, you should be able to see the list of customers:



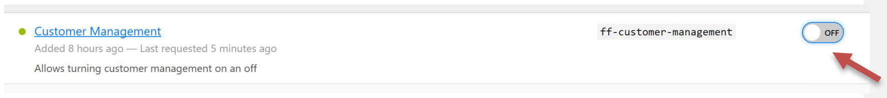
Customers

Create New

Id	Name	Address	
1	Joe Doe	100 main street	Edit | Details | Delete
2	Jane Doe	200 main street	Edit | Details | Delete
3	Jimmy Doe	300 main street	Edit | Details | Delete
4	Jesse Doe	400 main street	Edit | Details | Delete

© 2018 - ModernUIApp

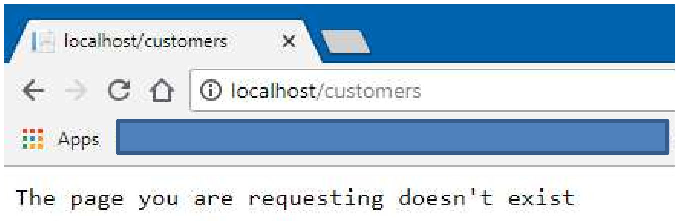
- Go to the Feature flag dashboard, and turn it off:



Customer Management
Added 8 hours ago — Last requested 5 minutes ago
Allows turning customer management on an off

`ff-customer-management`

- Refresh the customers page in your application. You should now get the following response:



The page you are requesting doesn't exist

You are now able to control the visibility of this feature directly from the launch darkly site by targeting specific users. Please note that in a real application, you will most likely get the user ids from the application itself instead of having them hardcoded, and you would use the JavaScript SDK as well so that the UI doesn't even offer a link to the page we requested if the feature is off.

Examine the code responsible of enabling the feature flag on and off. It is in the "CustomersController.cs" class in the "ModernUIApp" application:

```
public CustomersController(ICustomerDataService customerDataService,
IFeatureFlagService featureFlagService)
        {
            _customerDataService = customerDataService;
            _featureFlagService = featureFlagService;
        }
        public IActionResult Index()
        {
            if (_featureFlagService.ViewFeature(_userName))
            {
                var customers = _customerDataService.GetAll();
                return View(customers);
            }
            else
            {
                return Content("The page you are requesting doesn't exist");
            }
        }
```