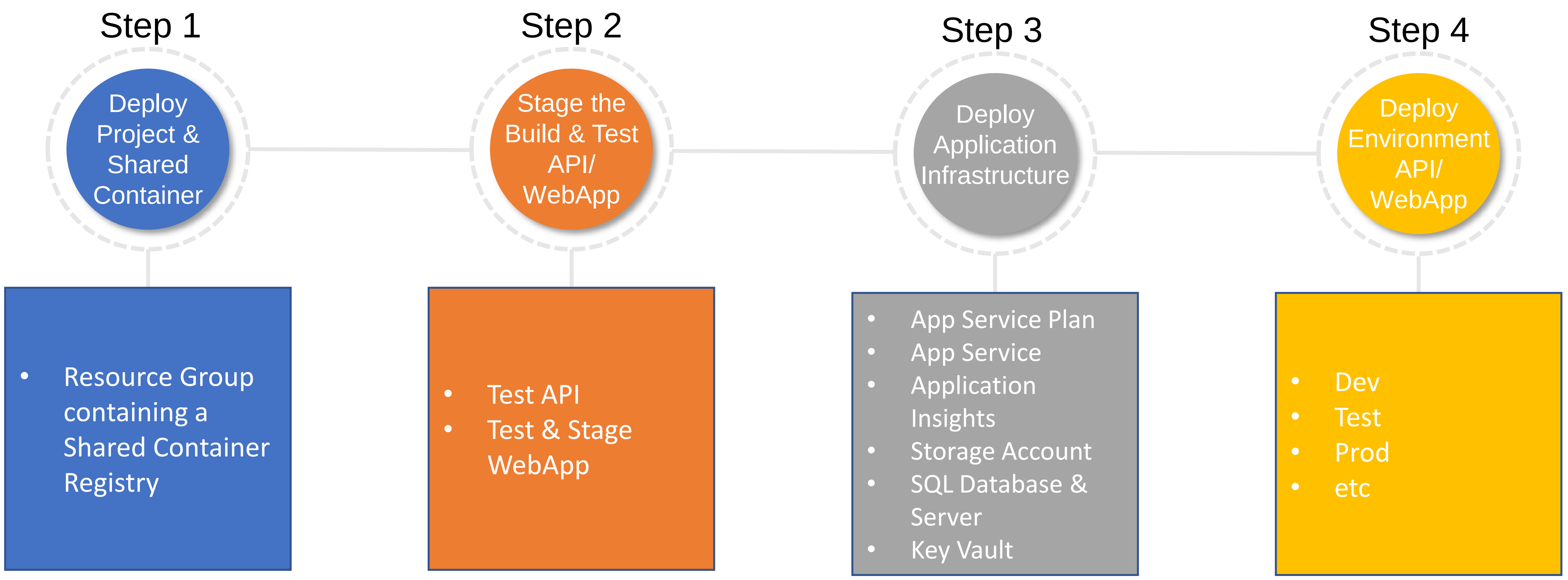
FastForge – GitHub Actions

2021

Pipeline Process Summary

The process required to start the FastForge project can be summarized in 4 main steps.


Step 1
Deploy Project & Shared Container
Step 2
Stage the Build & Test API/ WebApp
Step 3
Deploy Application Infrastructure
Step 4
Deploy Environment API/ WebApp

- Resource Group containing a Shared Container Registry

- Test API
- Test & Stage WebApp

- App Service Plan
- App Service
- Application Insights
- Storage Account
- SQL Database & Server
- Key Vault

- Dev
- Test
- Prod
- etc

Automation Process Summary (further explained)

Step 1

Deploy Project & Shared Container

GitHub

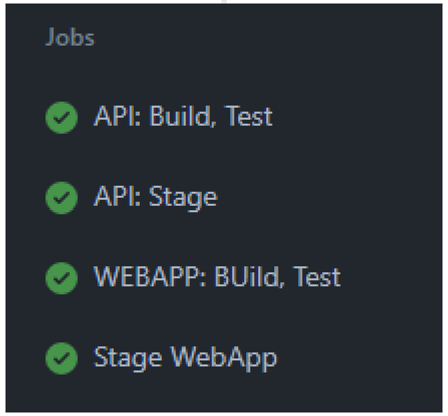
Azure Resource Groups

Azure Container Registry

1 Workflow/ file.yml

Step 2

Build Test & Stage API/ WebApp



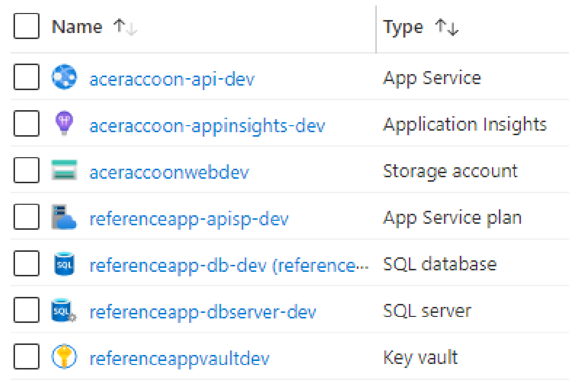
Jobs

- API: Build, Test
- API: Stage
- WEBAPP: BUild, Test
- Stage WebApp

1 Workflow / file.yml (per Environment)

Step 3

Deploy Application Infrastructure



Name ↑↓	Type ↑↓
aceraccoon-api-dev	App Service
aceraccoon-appinsights-dev	Application Insights
aceraccoonwebdev	Storage account
referenceapp-apisp-dev	App Service plan
referenceapp-db-dev (reference...	SQL database
referenceapp-dbserver-dev	SQL server
referenceappvaultdev	Key vault

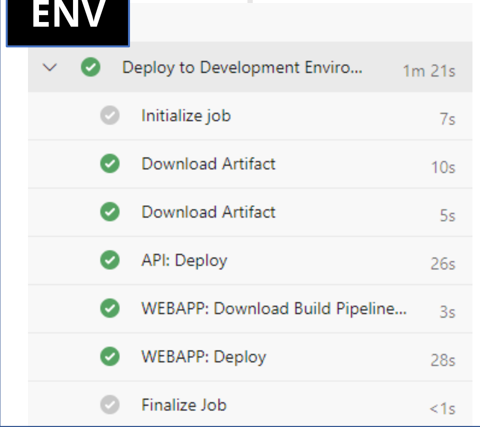
This step creates the Storage Account and other Resources needed for Step 4

1 Workflow / file.yml (per Environment)

Step 4

Deploy Environment API/ WebApp

ENV



Deploy to Development Enviro...	1m 21s
Initialize job	7s
Download Artifact	10s
Download Artifact	5s
API: Deploy	26s
WEBAPP: Download Build Pipeline...	3s
WEBAPP: Deploy	28s
Finalize Job	<1s

Starting Prerequisites for FastForge

- **Create/Re-Use Microsoft Azure Subscription**
    - The Azure Subscription will host all the required Infrastructure needed for FastForge
- **Create a GitHub Organization/Account**
    - GitHub will be used as the primary code repository and will manage the workflow for automation of deployment
- **Customize the App Dev Process Template for the Organization to be consistent with Centric's Right Site approach**
    - A key component of Centric's Right Site model is determining appropriate locations for our Client's project, files and folders that manage the FastForge deployment.
    - Create an inherited process derived from the existing Agile Template
    - Name the process and remember the name to be used later in the process
- **Determine early on whether the FastForge Repository will be private or public.**
    - Public repositories provide the option to use "Environments" for free, whereas a Private repository requires GitHub Enterprise
- Download and install the needed tools for deployment:
    - Azure CLI
    - GIT
    - Terraform Package
- **Establish a shared container within the Azure Subscription for the Terraform .tfstate file**
    - This step requires manually creating the following resources:
        - Resource Group
        - Storage Account
        - Container within Storage Account
            - The .tfstate file will be stored in this location
            - All subsequent .tfstate files will also be stored in this Storage Account Container

Step 1: Deploy Project & Shared Container

- The following resources and items will be created as part of step 1:
  - A Resource Group with a Container Registry located in the Azure Subscription
- To begin the creation of the foundational infrastructure required for the FastForge application, the owner of the GitHub Organization should perform the following steps:
  1. Navigate to the Centric GitHub repository and clone the contents onto your personal device (Note, save the folder location in a short folder path. Longer paths can cause terraform init to error):
     1. https://github.com/centricconsulting/FastForge-Foundation
  2. Open PowerShell and path to the folder “tf-GitHub” within the local repository location
  3. *Run the “az login” command to authenticate with the Azure Subscription
     1. Further specify the specific subscription using the following command:
        1. **az account set --subscription “<enter subscription name>”**
        2. **az account show**: Shows the current set account
  4. Adjust the main.tf file under the “backend” portion to specify the shared container location within the Azure Subscription
     1. **resource_group_name**: <Resource Group created in Prerequisites section>
     2. **storage_account_name**: <Storage Account created in Prerequisites section>
     3. **container_name**: <Container created in Prerequisites section>
     4. **key**: <tfstate file name for management of the created resources>
        1. (Optional, defaults to `GitHub-FastForge-Foundation.tfstate`)
  5. Update the “terraform.tfvars” file to change resource configurations for the Container Registry and Resource Group
  6. Save all file changes
  7. *Run the terraform commands in sequence:
     1. terraform init
     2. terraform plan --var-file="terraform.tfvars“
     3. terraform apply --var-file="terraform.tfvars“ –auto-approve

***Azure PowerShell modules, GIT, and the Terraform package must be installed on the device prior to running commands**

Step 2: **PUBLIC REPO -** Stage the Build & Test API/WebApp

- The Workflow to automate further resource deployments can run after the Container Registry is created. This step must be performed after step 1 in order to create a **public repository** for the FastForge build:
    1. Navigate to Centric's GitHub and fork the following repository into your Organization's GitHub.
        1. https://github.com/centricconsulting/FastForge-ReferenceApp-dotnet.git
    2. Navigate to the "Actions" tab within your Organization's GitHub main page and enable workflows for your repository
    3. Navigate to the "Settings" tab within your Organization's GitHub main page and create "secrets" needed in order to connect to the previously provisioned Azure Container Registry. The variable information can be found under the "Access Keys" section of the Container Registry.
        1. **REGISTRY_LOGIN_SERVER**: <<Name of Container Registry>.azurecr.io>
        2. **REGISTRY_USERNAME**: <Chosen Username>
        3. **REGISTRY_PASSWORD**: <Chosen Password> (Be sure to adjust this variable if the password is refreshed on the Container Registry)
    4. Once the above secret values are created, the first workflow can run:
        1. Open the "BuildTestStage-apiWebApp.yml" file and ensure the ref path within the .yml file is pointing to the "main" branch if applicable
        2. Navigate to the "Actions" tab within your Organization's GitHub main page:
            1. Select the "BuildTestStage-apiWebApp (run 1st)" workflow, and select "Run workflow" on the right-hand side

Step 2: **PRIVATE REPO** - Stage the Build & Test API/WebApp

- The Workflow to automate further resource deployments can run after the Container Registry is created. This step must be performed after step 1 in order to create a **private repository** for the FastForge build:
  1. Navigate to your organization's GitHub and create a new empty private Repository
  2. After the repository is created, import Centric's GitHub repository for the FastForge deployment:
     1. https://github.com/centricconsulting/FastForge-ReferenceApp-dotnet.git
  3. Navigate to the "Actions" tab within your Organization's GitHub main page and verify there are 3 different workflows. If you do not see a specific workflow, navigate to the workflows (.github/workflows/) and make a small edit to the .yml file, such as adding a blank space after a word. You should now see the workflow present under the "Actions" tab.
  4. Navigate to the "Settings" tab within your Organization's GitHub main page and create "secrets" needed in order to connect to the previously provisioned Azure Container Registry. The variable information can be found under the "Access Keys" section of the Container Registry.
     1. **REGISTRY_LOGIN_SERVER**: <<Name of Container Registry>.azurecr.io>
     2. **REGISTRY_USERNAME**: <Chosen Username>
     3. **REGISTRY_PASSWORD**: <Chosen Password> (Be sure to adjust this variable if the password is refreshed on the Container Registry)
  5. Once the above secret values are created, the first workflow can run:
     1. Open the "BuildTestStage-apiWebApp.yml" file and ensure the ref path within the .yml file is pointing to the "main" branch if applicable
     2. Navigate to the "Actions" tab within your Organization's GitHub main page :
        1. Select the "BuildTestStage-apiWebApp (run 1st)" workflow, and select "Run workflow" on the right-hand side

Step 3: Deploy Application Infrastructure for each <env>

- The following steps deploy the needed application infrastructure within the Azure Subscription for a single environment:
  1. In order to access the Azure Subscription, 5 additional secrets need to be created. These 5 secrets are required by the Terraform files for the GitHub Actions agent to access the Azure Subscription.
     1. **AZURE_CREDENTIALS**: <Process of creation>
        1. Run the following command in PowerShell:
           1. az ad sp create-for-rbac --name "{sp-name}" --sdk-auth --role contributor --scopes /subscriptions/{subscription-id}
           2. Run: az ad sp create-for-rbac and paste the entire value as the secret. Some values placed here will need to be their own separate Secret as shown in items **2-5** below
     2. **CLIENT_ID**: <Client ID of Azure credentials created above>
     3. **CLIENT_SECRET**: <Client Secret of Azure credentials created above>
     4. **SUBSCRIPTION_ID**: <Subscription ID for the Azure Subscription>
     5. **TENANT_ID**: <Tenant ID for the Azure Subscription>
  2. Navigate to the "Settings" tab within your Organization's GitHub main page and select "Environments". Select "New environment" and name the environment based on the working environment. (*Note, this step is available for Public repositories, but requires a GitHub Enterprise license for Private repositories*)
     1. Add a reviewer for this step and place up to 6 resources that will approve infrastructure deployments. This step will force manual intervention to occur between the "Terraform Plan" and "Terraform Apply" stages within the workflow.
     2. **If this step is performed**, be sure to uncomment and include lines 70-71 in the < env>-infrastructure.yml file with the appropriate environment.
  3. Update the "terraform.tfvars" file for the appropriate environment (/tf-infrastructure/<env>-env) file with the required values for deployment
  4. Open the <env>-infrastructure.yml file for the appropriate environment and adjust the "env" variables to reflect where the tfstate file will be located for the Terraform managed resources.
     1. **resourceGroup**: <Name of Resource Group created in prerequisites section>
     2. **storageAccountName**: <Name of Storage Account created in prerequisites section>
     3. **storageContianerName**: <Name of container created in prerequisites section>
     4. **storageKey**: <Name of tfstate file. Must be in "<name>.tfstate" format. Make the name reflect the environment>
  5. Save/commit all changes and follow the process outlined in Step 2, but select the "Dev-infrastructure (run 2nd)" workflow to run

Step 3 continued: Created Resources

- After the deployment of Step 3 is complete, the following resources should be created in your Azure Subscription:
    1. Resource Group: **<app_name_[environment]>**
    2. App Service Plan: **<app_name-apisp-[environment]>**
    3. App Service: **<app_name>**
    4. SQL Server: **<app_name-dbserver-[environment]>**
    5. SQL database: **<app_name-db-[environment]>**
    6. Key Vault: **<app_name-[environment]-kv01>**
    7. Storage account: **<app_nameweb[environment]>**
    8. Application Insights:**<randomName-appinsights-[environment]>**

Step 4: Deploy Environment API/WebApp

- The last step in the build of FastForge is the deployment of each environment’s API and WebApp. The following steps are required to complete the deployment:
  1. Use the “env” section within the .yml file (<env>-deploy-apiWebApp-pipeline.yml) to store the following variable:
     1. **apiAppName**: <Name of app service created in Step 3 >
  2. Save/commit all changes in the .yml file
  3. Navigate to the “Settings” tab within your Organization’s GitHub main page and create the last “secret” needed in order to connect to the storage account that was provisioned in Step 3
     1. **AZURE_STORAGE_ACCOUNT_CS:** <Connection String for the Storage Account>
  4. Navigate to the “Actions” tab within your Organization’s GitHub main page:
     1. Select the “<env>-deploy-apiWebApp.yml” workflow, and select “Run workflow” on the right-hand side
        1. After running the workflow, the API/WebApp will be built for the <env> specified
  5. When needed, repeat the above steps to deploy the API/WebApp for each subsequent environment. Step 3 must occur first for each environment in order to have the resources required to deploy Step 4.