Modern AppDev Workshop Student Guide:

Table of Contents

[Workshop Overview & Timeline 2](#_Toc489529454)

[Module #1 – Installation 3](#_Toc489529455)

[Module #2 – Environment Verification 6](#_Toc489529456)

[Module #3 – Modifying the Game Images 11](#_Toc489529457)

[Module #4 – Adding End User Experience with HEC 16](#_Toc489529458)

[Module #5 – HEC in High Volume Environments 18](#_Toc489529459)

[Module #6 – Adding an APM Tool - New Relic 19](#_Toc489529460)

[Module #7 – Examining Gaming (Business) Metrics 21](#_Toc489529461)

[Module #8 – Feedback About the Session 21](#_Toc489529462)

[Appendix – Notes and Helper Files 22](#_Toc489529463)

# Workshop Overview & Timeline

**Module #1 – Installation 15 minutes**

* Git Installation for OSX, Windows and Linux
* Docker Installation
* Start the Docker cluster

**Module #2 – Environment Verification 25 minutes**

* Play the Buttercup-Go game
* Verify your Splunk setup
* Verify your Bitbucket setup
* Verify your Jenkins setup
* Clone your Bitbucket repository to your local machine

**Module #3 – Modifying the Game Images 40 minutes**

* Creating a Git Branch & Checkout for Images
* Replace the color images with gray scale images
* Adding, Committing and Pushing the new Images
* Invoke the Build Plan for the Images
* Verify Changes via Web Browser

**Module #4 – Adding End User Experience with HEC 20 minutes**

* Creating Git Branch, Check, and Push for HEC
* Verify Changes via Splunk

**Module #5 – Adding an APM Tool - New Relic 20 minutes**

* Creating Git Branch, Check and Push for New Relic
* Verify Changes via Splunk

**Module #6 – Adding an Easter Egg 20 minutes**

* Creating Git Branch, Check, and Push for Fast Mode
* Verfy Changes via Web Browser or Mobile Phones

**Module #7 –Examining Gaming (Business) Metrics 20 minutes**

* Explore the Buttercup Go App in Splunk
* Explore the Jenkins App in Splunk
* Explore the End User Experience App in Splunk
* Explore the New Relic App in Splunk

**Module #8 –Feedback 5 minutes**

* Feedback About the Session

# Module #1 – Installation

In this module, we will ensure that everything is installed and setup correctly.

1. Create a 'workshop' directory on your local machine.
   1. Open a command prompt/terminal session
   2. Create a directory named workshop  
       mkdir workshop
   3. Change directories to your newly created workshop directory  
       cd workshop
2. Download the workshop materials
   1. Download from here: <https://github.com/tmartin14/modern-app-dev-workshop>  
      Click the green 'Clone or download' button and choose **Download ZIP** (~15MB)
   2. Extract the workshop materials into the \workshop directory you created earlier
   3. VERIFY your working directory is titled: **modern-app-dev-workshop-master**
   4. The workshop materials include a links.html file that you can open and use for quick reference, but we'll include the links here as well:

Buttercup-Go <http://localhost:3040>

Splunk Enterprise <http://localhost:8000>

Bitbucket <http://localhost:7990>

Jenkins <http://localhost:8080>

1. Install Git (git command-line binaries)
   1. **Mac OSX Installation of git using Homebrew (**https://brew.sh/**):**

Open your terminal and install git with Homebrew:  
 brew install git  
**-OR-**

Download the pkg and install it from :  
<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

* 1. **Windows Installation of Git:**

Download the latest version of Git for Windows (<https://git-for-windows.github.io/> ) & Run the Setup

* 1. **Linux Installation of Git:**

For Ubuntu: sudo apt-get install git  
 or Redhat: dnf install git   
 -or- yum install git

1. Verify the version of Git installed is higher than 1.8:  
    git version
2. Open a Command Prompt and configure your Global Git Name and Email:  
    git config --global user.name "Your Name"  
    git config --global user.email [you@yourcompany.com](mailto:you@yourcompany.com)
3. Install Docker & docker-compose
   1. Goto <https://www.docker.com>
   2. At the top of the page click the link titled "Get Docker 🡪 For Desktops"
   3. Select your desktop OS and follow the links/instructions to install
   4. Verify that you are running at least version 17.05.0-ce of docker  
       docker –version
   5. Verify that you are running at least version 1.11.1, build 7c5d5e4 of docker-compose  
       docker-compose –version

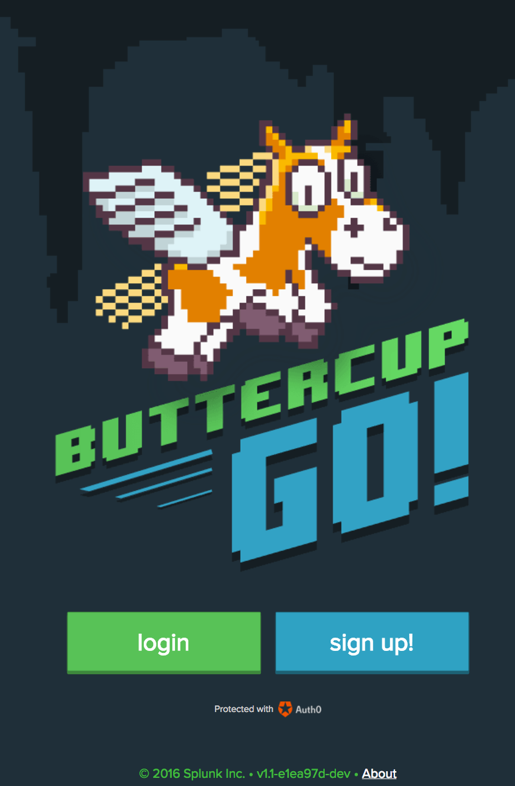
*Note: If you have any trouble installing docker or docker-compose, please check the list on known issues in the* [*Appendix*](#_Appendix_A_–) *at the end of this document.*

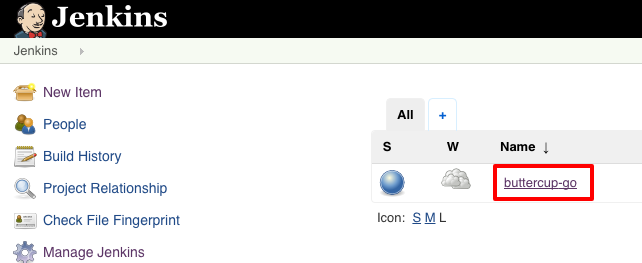
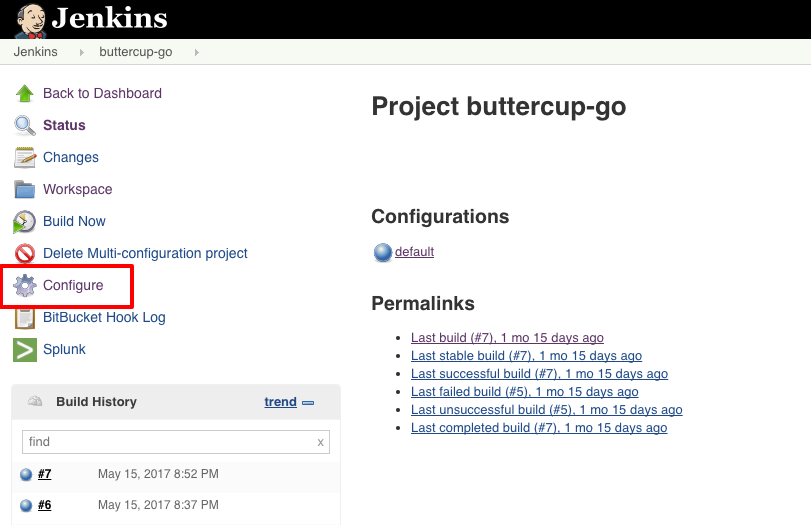
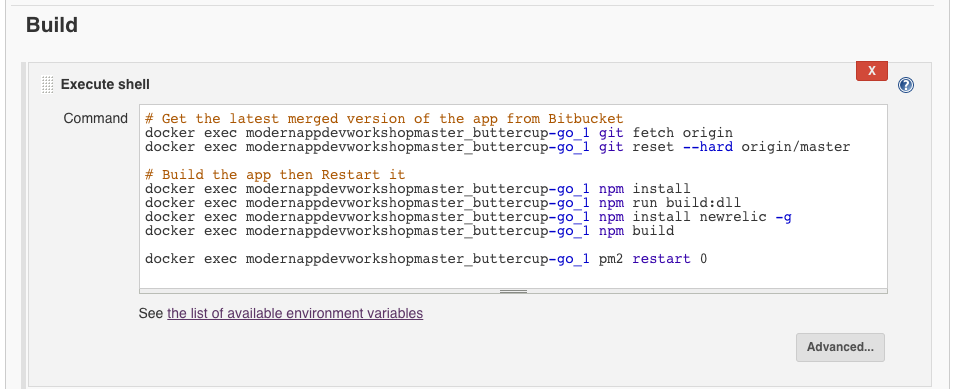
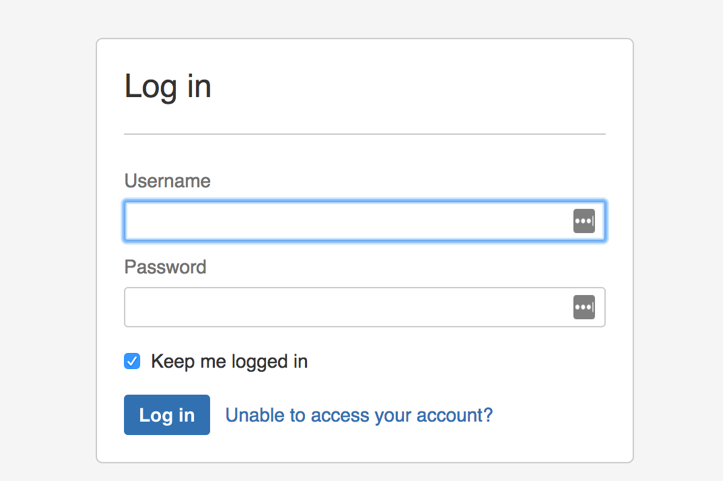
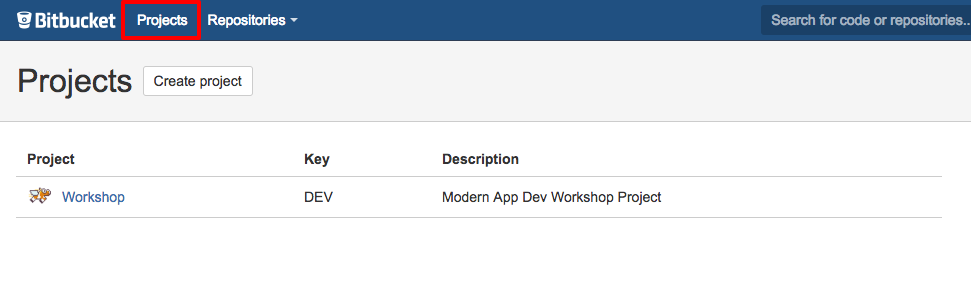
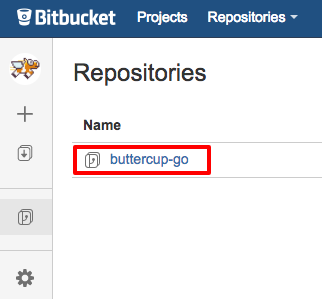
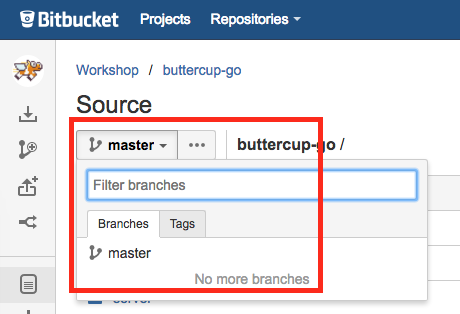
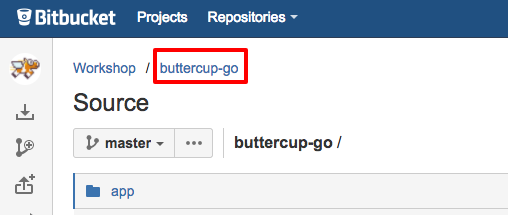
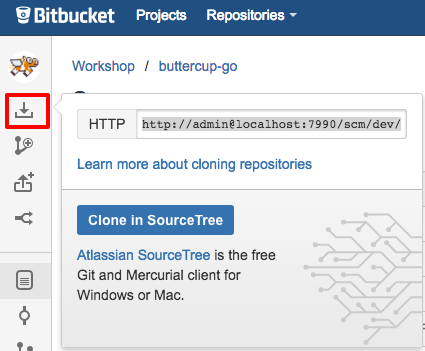
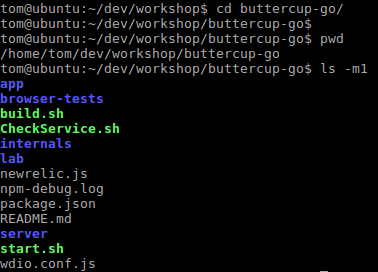
*\*\*\* IMPORTANT NOTE \*\*\*:   
For the next step please make sure you are in the* /modern-app-dev-workshop-master *directory when executing these commands. If not, the Jenkins build plan will not work.*

1. **Start the workshop environment!**
   1. Start Docker
   2. Ensure docker has enough space by executing:   
       docker system prune
   3. Initialize Docker Swarm (our environment runs in a cluster) docker swarm init
   4. Pull the latest docker images for this workshop  docker-compose pull
   5. Start the docker cluster for this workshop  docker-compose up -d
   6. Verify you have access to your docker container:  
       docker exec modernappdevworkshopmaster\_buttercup-go\_1 ls  
      This should produce a directory listing.

That's it! It will take a few minutes (5-10 minutes, be patient) for everything to spin up, but your cluster should now be up & running!

# Module #2 – Environment Verification

* 1. Verify your Node.js environment is up and running
  2. Goto <http://localhost:3040>
  3. Login using Google, Facebook, LinkedIn, or Twitter  
     ****
  4. Play Your Game!

1. Verify your Splunk login
   1. Goto <http://localhost:8000>
   2. Login with username: **admin** password: **changeme**
   3. Verify you have access and can navigate through some of the apps & dashboards
2. Verify you can access Jenkins and review your build plan
   1. Goto <http://localhost:8080>
   2. Login with username: **admin** password: **changeme**
   3. Confirm you have access then select your buttercup-go project.   
      
   4. From the buttercup-go project and select the “Configure” link from the left-hand menu.   
      
   5. Scroll to the bottom of the page you will see the actual steps in the build plan. The plan will:  
       1. Fetch the latest version of the master repo into your buttercup-go container  
       2. Install node modules   
       3. Rebuild the application  
       4. Restart the application  
        
      This script will execute every time your project builds.  
      
3. Verify your Bitbucket repository (repo) is accessible and setup properly
   1. Goto <http://localhost:7990>
   2. Login with username: **admin** password: **changeme**
   3. Confirm you have access and navigate to the Projects tab. 
   4. Click the Workshop Project and then select your buttercup-go repo.  
      
   5. You should see only a 'master' branch in the dropdown list.  
        
       (This is the repo that you will be cloning to your laptop later)
4. Clone your Bitbucket repository to your local machine's /workshop directory
   1. Goto <http://localhost:7990>
   2. Select your repo  
      
   3. Click the Clone button on the left menu and **copy** the clone URL:  
      
   4. Open a command prompt/terminal session &   
      ensure you are in the \workshop directory you created earlier.
   5. Clone the master branch of your repo to your local machine  
       git clone http://admin:changeme@localhost:7990/scm/dev/buttercup-go.git  
        
      This will create a buttercup-go directory in your workshop directory. Your source code will be located at workshop\buttercup-go
   6. Change directories into your source. List the contents of your buttercup-go directory.   
       cd workshop\buttercup-go  
       ls –m1  
        
      Your listing should look similar to this:  
      

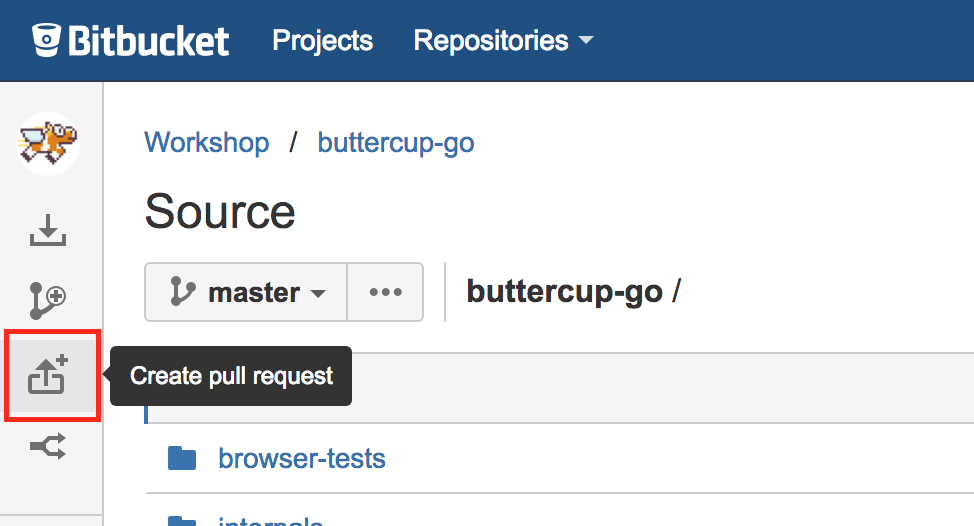
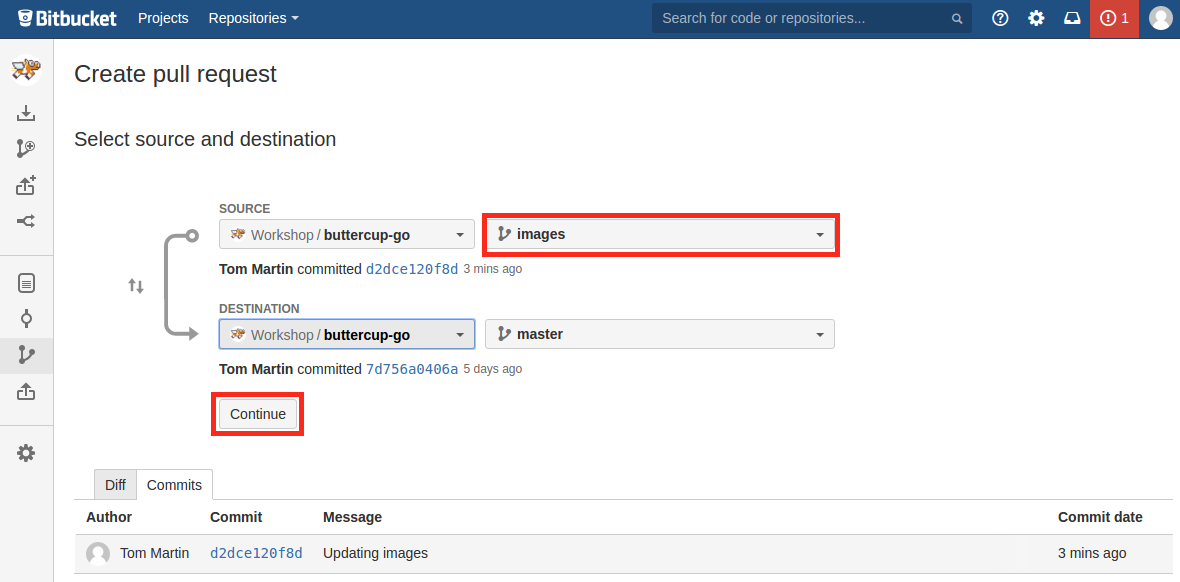
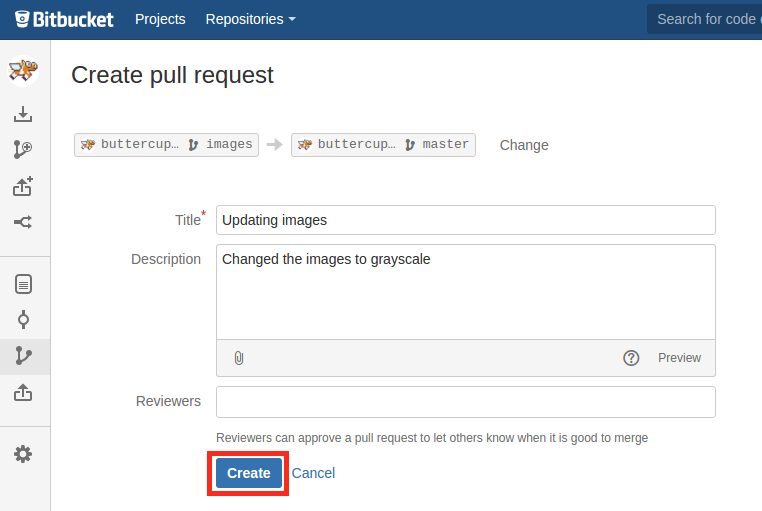
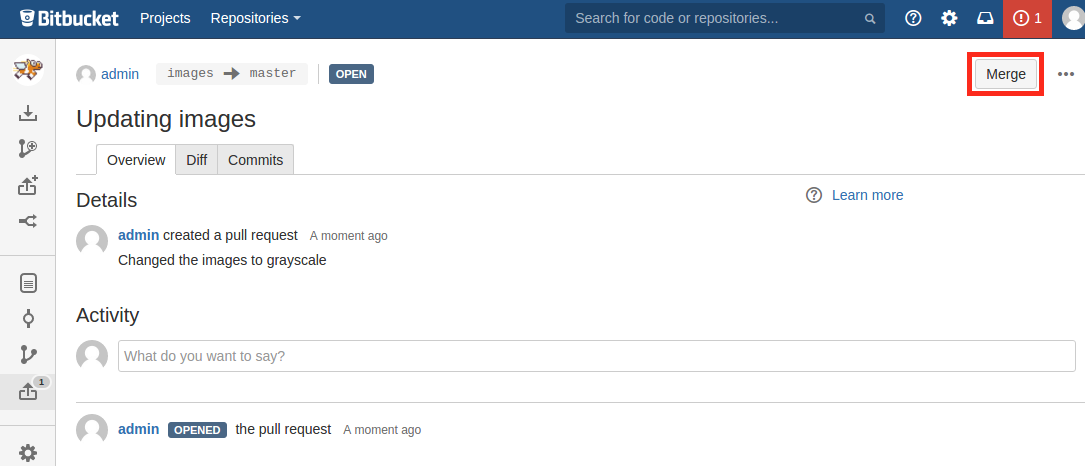
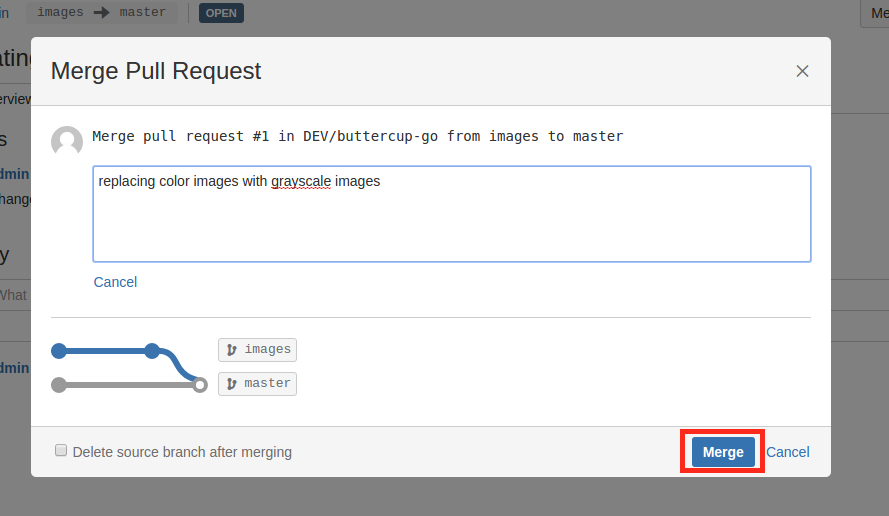
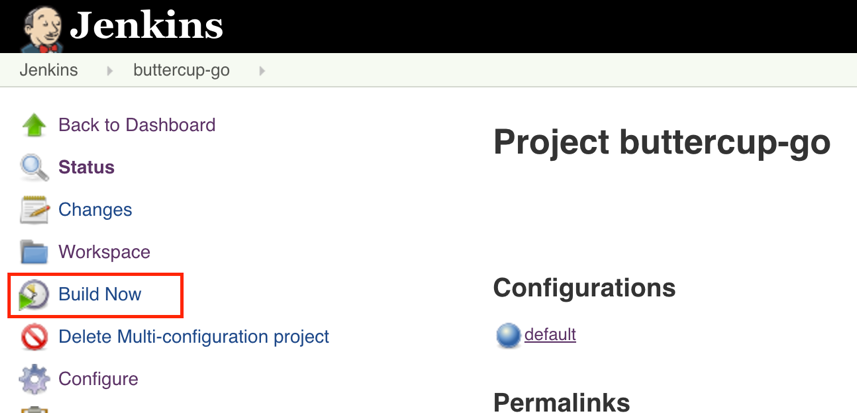
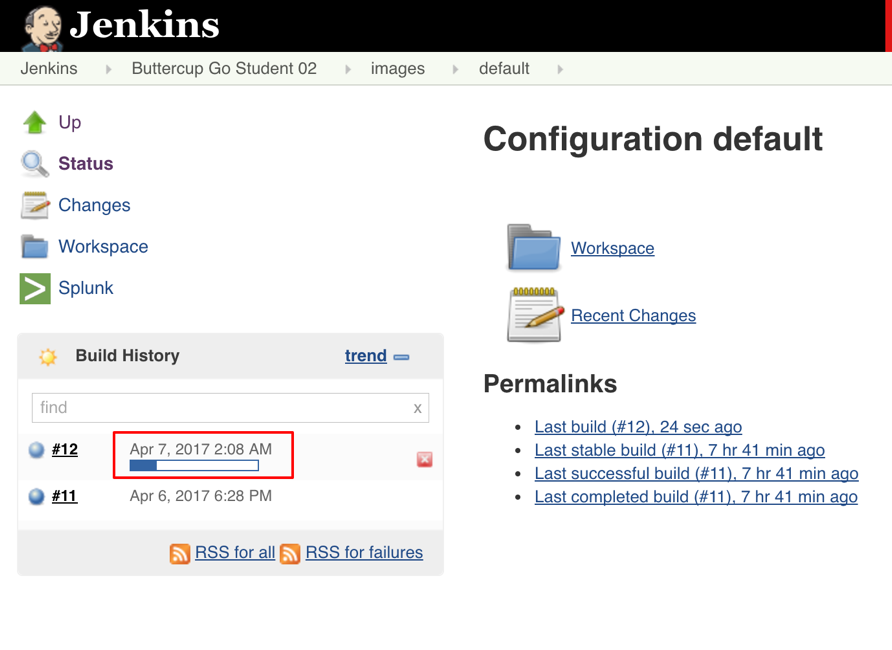
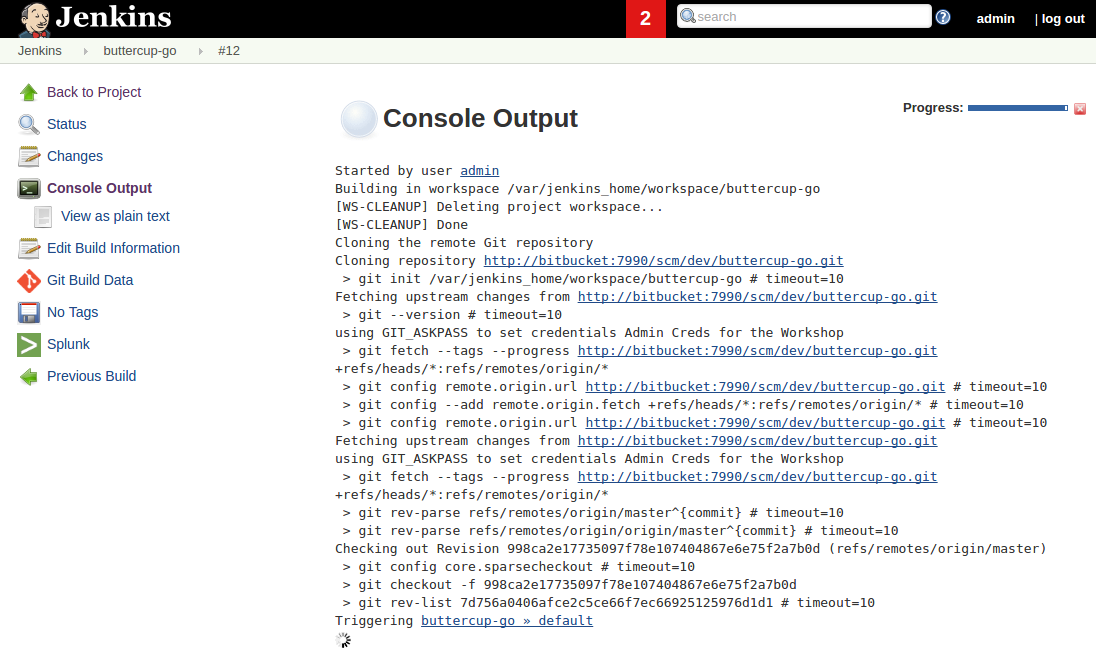
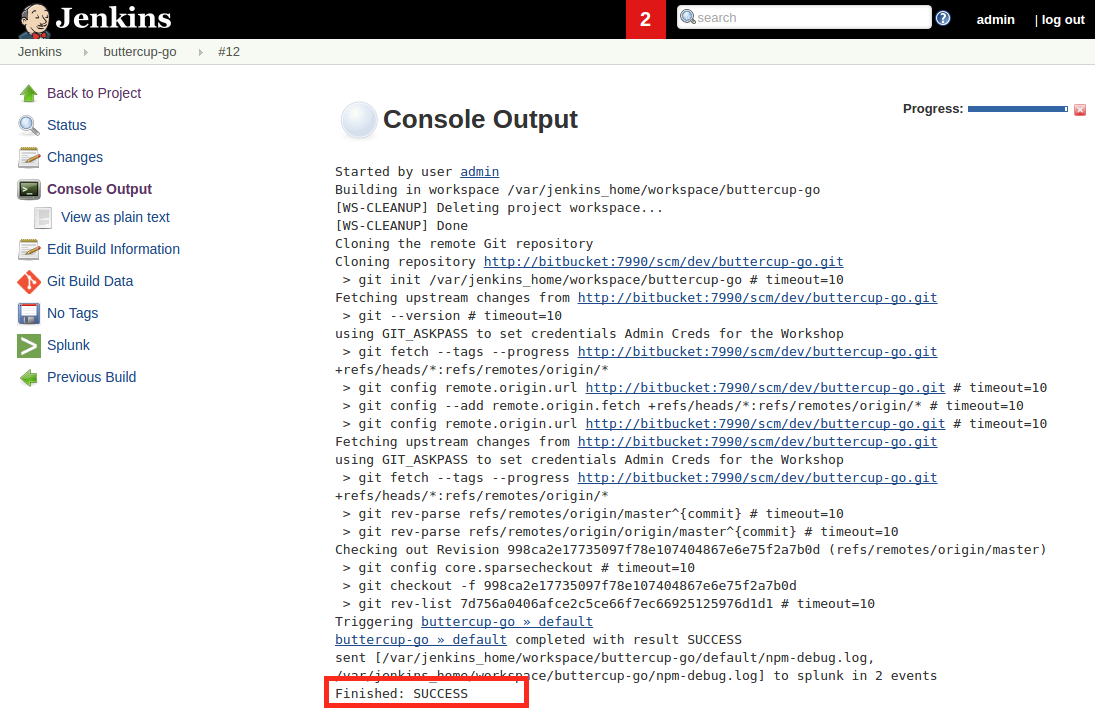
We’re now setup to start contributing code to the project!

# Module #3 – Modifying the Game Images

In this module, you will replace the colorful images in the game with a whole new set of grayscale images. You will create a new “branch” using git, check that branch out, move files from 1 directory to another, commit them and push your new files to Bitbucket. Let’s get started.

1. Make the code changes and push them to Bitbucket
   1. From the command line, create a new branch in your repo for these changes.  
       git branch images
   2. Now set this as our working branch   
       git checkout images
2. Now let's change the set of images & sounds the application will use.
   1. Copy the grayscale images over the original images in the app/assets directory
      1. IN THE buttercup-go DIRECTORY, execute:  
          cp -Rf app/assets-gray/. app/assets/

This will copy the grayscale images into the assets directory where they will be picked up at run time. That's it! Code changes are complete.

1. Add our changes to our images branch  
    git add \*
2. To see a summary of the changes you've just made to the codebase, let's check the status  
    git status
3. Commit those changes to our branch   
    git commit   
   You will need to supply a commit message, use something like "Updating Images".
4. Finally, let’s push the changes and we’re done!   
    git push origin images
5. Goto Bitbucket and create a pull request for your changes  
    http://localhost:7990   
   1. Create a Pull Request:  
      ****
   2. Select your images branch:  
      ****
   3. Create the Pull Request ****
   4. Now merge our changes into the master branch:  
      ****
   5. Add your comments and your code will be merged!  
      ****
6. **Goto Jenkins and verify the build plan for the images branch executed successfully.** 
   1. Goto Jenkins: http://localhost:8080
   2. Navigate to your buttercup-go project & click the "Build Now" link  
      (There is a bug in the current Jenkins plug-in for Bitbucket 5.x that prevents the build from triggering automatically so we need to manually force it).  
      ****
   3. Verify your build plan executed successfully:
      1. Select your “default” build process in the list, the click the progress bar in the resulting window. 
      2. That will show the console output from your build process  
          ****
      3. Once the build is complete, you should see the following message:  
           
           
         If you want to see more details logs, check the 'Console Log' for the default plan:   
          http://localhost:8080/job/buttercup-go/default/XX/console
7. **Did your build fail?** If so, please make sure that the directory your code is in matches the name in the Build Plan. Your directory should be \modern-app-dev-workshop-master which is represented as modernappdevworkshopmaster\_buttercup-go\_1 in the build plan. If your directory is different, just update your build plan.
8. **Once the build completes, play your game again and notice the updated images.**   
     
   *Note: If the build fails due to a timeout, you can adjust the timeout in the Jenkins build plan. This could happen if you're on a machine with less than 16 GB of RAM or if you're internet connection is slow (the build will pull down a lot of node.js libraries).*

# Module #4 – Adding End User Experience with HEC

In this module, you will add code to measure the End User’s Experience with your application using the open source library, Boomerang.js. You will create a new “branch” using git, check that branch out, modify files, commit and push you changes to Bitbucket. Let’s get started.

1. Make the code changes and push them to Bitbucket
   1. From the command line, create a new branch in your repo for these changes and set this as our working branch.  
       git branch hec  
       git checkout hec
   2. Edit the file buttercup-go/app/index.html adding the following code just above the </head> tag:

<!-- Boomerang.js Browser Code -->

<script src="/scripts/boomerang/boomerang.js"></script>

<script src="/scripts/boomerang/plugins/rt.js"></script>

<script src="/scripts/boomerang/plugins/navtiming.js"></script>

<script src="/scripts/boomerang/plugins/guid.js"></script>

<script src="/scripts/boomerang/plugins/mobile.js"></script>

<script>

BOOMR.init({

beacon\_url: "http://localhost:8088/services/collector/raw?channel=00000000-0000-0000-0000-000000000003",

beacon\_type: "POST",

beacon\_auth\_token: "Splunk 00000000-0000-0000-0000-000000000003",

site\_domain: "Buttercup-Go"

});

BOOMR.addVar({ "ua\_raw" : navigator.userAgent ,

"appName" : "Buttercup-Go" });

</script>

* 1. Now lets add our changes to the repo, commit them, and push the changes

git add \*

git commit (use: "Adding End User Experience monitoring")

git push origin hec

* 1. Goto Bitbucket and create a pull request for your changes
  2. Merge your changes into the master branch
  3. Goto Jenkins and execute the build plan for the hec branch
  4. Verify your changes are now in the buttercup-go application:  
     docker exec modernappdevworkshopmaster\_buttercup-go\_1 cat app/index.html

1. Play your game again. Get as far as you can.
2. Login to Splunk ( <http://localhost:8000> ) and examine the “End User Monitoring” application. You should be able to see the page performance in the App in Splunk. Can you explain the difference between the TTFB and Page Load times?

# Module #5 – HEC in High Volume Environments

The Splunk HTTP Event Collector is designed to handle high volume traffic. In this module we will use a hidden feature of the Buttercup-Go game to insert many logging events via the HEC REST endpoint.

1. Reload your game and click the 'About' link in the footer
2. Click the .conf link /Users/tomm/Desktop/conf2017a.png (Hey, why not register while you're there?)
3. Click the back button and play again. Notice anything different?
4. Login to Splunk and see how many more events are flowing into HEC. Have your scores improved? ;-)

# Module #6 – Adding an APM Tool - New Relic

In this module, you will add code required to add a New Relic agent to the Node.js application (/server) and a New Relic Javascript snippet to track browser behavior to the applications (/app). You will follow the same process for code checkout/commit as the prior labs. You will create a new branch using git, check that branch out, modify files, commit and push you changes to Bitbucket. Let’s get started.

1. Create a new branch for our New Relic changes
   1. From the command line, create a new branch in your repo for these changes and set this as our working branch  
       git branch new\_relic  
       git checkout new\_relic
2. Add the New Relic APM Agent to your node.js **server** application
   1. Create a new file in the root directory buttercup-go/newrelic.js that reads as follows:

'use strict'

/\*\*

\* New Relic agent configuration.

\*

\* See lib/config.defaults.js in the agent distribution for a more complete

\* description of configuration variables and their potential values.

\*/

exports.config = {

/\*\*

\* Array of application names.

\*/

app\_name: ['Buttercup-Go-**something-unique**],

/\*\*

\* Your New Relic license key.

\*/

license\_key: 'eacafcaa410da96769973eef817e765d1d6a39e5',

logging: {

/\*\*

\* Level at which to log. 'trace' is most useful to New Relic when diagnosing

\* issues with the agent, 'info' and higher will impose the least overhead on

\* production applications.

\*/

level: 'info'

}

}

* 1. Add a reference to the newrelic.js file to your Node.js server application. Edit the file buttercup-go/server/index.js adding the following code **as the very first line** of the file**.** require('newrelic');

1. Add New Relic Browser code to your web application's HTML.
   1. Edit the file buttercup-go/app/index.html adding the following code just above the </head> tag in the file.

<!-- BEGIN New Relic Browser Code -->

<script src=" /scripts/newrelic/newrelic.js"></script>

<!-- END New Relic Browser Code -->

1. Now let's add our changes to the repo, commit them, and push the changes.

git add \*

git commit (use: "Adding New Relic monitoring")

git push origin new\_relic

1. Goto Bitbucket and create a pull request for your changes
2. Merge your changes into the master branch
3. Goto Jenkins and execute your build plan for the new\_relic branch
4. Verify your changes are now in the buttercup-go application:  
   docker exec modernappdevworkshopmaster\_buttercup-go\_1 cat app/index.html
5. Play your game again. Get as far as you can.
6. Login to Splunk and examine the “New Relic" Application. You should be able to see New Relic data for your application in the dashboard within Splunk.   
     
   How does your application compare to those of your classmates?

# Module #7 – Examining Gaming (Business) Metrics

In this module, you will explore some of the dashboards and searches used to create those dashboards throughout this workshop. The goal is for to see how easy it is to visualize, correlate, search and discover new insights about your modern application development environment.

1. Login to Splunk
2. Explore the Buttercup Go App in Splunk
3. Explore the Jenkins App in Splunk
4. Explore the End User Experience App in Splunk
5. Explore the New Relic App in Splunk
6. Create a dashboard of your own that brings together components of each of the above Apps to provide a DevOpsApm view of Buttercup-Go
7. Create a New Dashboard by visiting the Search and Reporting App
   1. Create a search using index=buttercupgo
   2. Save as a Dashboard Panel and name you dashboard.
   3. Continue adding additional panels for

index=boomerang

index=apm

index=Jenkins

Be creative, we might keep your dashboard. User Real time or historical time settings. Have Fun!

# Module #8 – Feedback About the Session

Thank you for taking this DevOps journey with us. If you have any questions please feel free to contact us via twitter or contact your local Splunk account team

Please help us make this workshop better by completing this simple survey (less than 5 minutes):

<https://www.surveymonkey.com/r/35LZ785>

ALL feedback is welcome, positive or areas for improvement.

# Appendix – Notes and Helper Files

**Docker Cleanup**

After completing this workshop you will want to shutdown the environment and optionally remove all of the containers rom your local machine. These commands will help.

1. **Stopping the workshop environment**   
    docker-compose down  
     
   **\*\*\* Warning: All Changes will be lost when you stop the cluster! \*\*\***

**Workshop Reset & Helper Files**

In the repo for the buttercup-go application you will notice there is a "lab" folder. This folder contains two sets of files and a shell script to help you reset your environment should you need to. The reset\_files.sh script will set the environment to either the starting point or the completed point of the workshop should you need a "reset".

You can execute the script and restart your application by issuing the following command (all on 1 line):

docker exec modernappdevworkshopmaster\_buttercup-go\_1 bash -c "cd lab && ./reset\_files.sh <starting>|<final> && pm2 restart 0"

Please note the option to specify <starting|final> when executing this script.

You can also use the individual files in the starting\_files and ending\_files directories if you need an example of the changes to be made (cut-and-paste is your friend ;-). Feel free to use any of these if you're having troubles with the labs.

**Docker Installation**

There are a few known issues with Docker that can be addressed fairly easily. Here are some that we're aware of already:

1. **Older Version of Windows** - If you're running an older version of Windows (like Windows 7), you may need to use the Docker Toolbox. Please check here for notes on running docker for older OS's: <https://docs.docker.com/toolbox/toolbox_install_windows>
2. **Linux issue with docker-compose** – There is a known bug in the linux version of docker-compose where it will not recognize docker-compose.yml files written in version 3.x. Fortunately there is a published fix for this issue and we've included a shell script to implement that fix. The steps to resolve this issue are:  
   1. Remove docker-compose from you system if it's installed.  
       apt-get remove docker-compose  
       -or-  
       sudo rm /usr/local/bin/docker-compose
   2. Run the install script  
       sudo ./install-docker-compose.sh

Once you've done this you should be running version 1.11.1 build 7c5d5e4 of docker-compose and you're all set.

**npm known version issues**

Note: This only applies if you're trying to run the app on your local machine. This has already been configured in the buttercup-go docker image.

The latest version of npm for node has some known bugs so this workshop uses a stable version of npm, version 4.6.1. The workshop has been tested and works with this version. If you're environment has a different version, please update it to version 4.6.1 with this command:

npm install -g npm@latest-4