



'Git' Splunk App Management with Visual Studio Code

Erica Pescio & Joe Welsh

Forward Deployed Software Engineering | Splunk



Erica Pescio

Forward Deployed Software Engineering | Splunk
epescio@splunk.com



Joe Welsh

Manager, Forward Deployed Software
Engineering | Splunk
jwelsh@splunk.com

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Listen to Your Data, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2019 Splunk Inc. All rights reserved.

Challenges

What causes heartburn when developing and managing
Splunk apps



Developing & Managing Splunk Apps

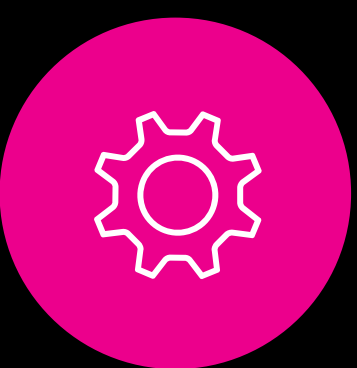
Doesn't Come Without It's Challenges



No “Real” Splunk Config
File Editor



Splunk Lacks File
Versioning



No Single Interface or IDE

Impact

Implications of doing it the same old way



Time Is Valuable, Errors Are Costly, Mistakes Happen

- ▶ Syntactical errors cost development cycles
- ▶ Editing remote files with basic text editors cumbersome and error prone
 - One alternative is to transfer files to local environment, edit, and push back
- ▶ No way to keep track of changes made within your Splunk application (or environment)
- ▶ No Single Development Interface or Integrated Development Environment

Solution

Using GitLab and Visual Studio Code to develop and manage your Splunk Applications



GitLab, Visual Studio Code & Splunk

“The Whole is Greater Than The Sum Of Its Parts”?



GitLab

- Online Code Storage
- Collaborative Environment
- Version Control
- Inspect Previous Code
- Roll Back Changes
- Free Hosting and Services



Visual Studio Code

- Source Code Editor
- Cross Platform
- Splunk Code Completion
- Splunk Syntax Correction
- Integrated Terminal
- Free To Download



Splunk

- Structured, Text-Based Configuration Files
- Structured Configuration File Storage
- Command Line Interface

How They Work Together

Visual Studio Code Integrates Everything You Need



Splunk

Splunk Enterprise

A Great, Structured & Modular File System But Not Without It's Challenges

A single Splunk instance can have [multiple versions of configuration files across several directories](#).



Splunk does not have

- ▶ Automatic file backup
- ▶ Version control
- ▶ No “True” File Editor

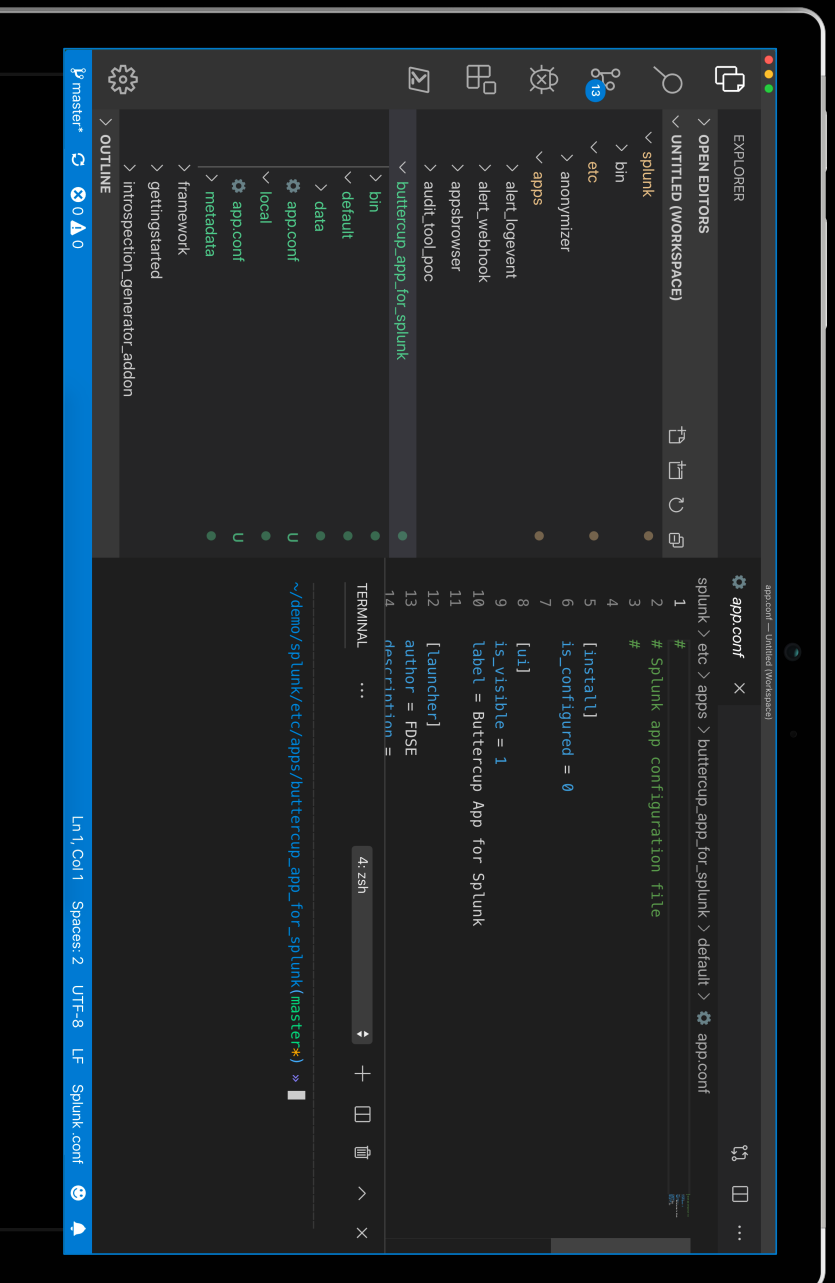
Splunk does have

- ▶ A single directory that contains your apps files
\$SPLUNK_HOME/etc/apps/<your_app>

Visual Studio Code

Light-Weight, Cross Platform, General Source Code Editor

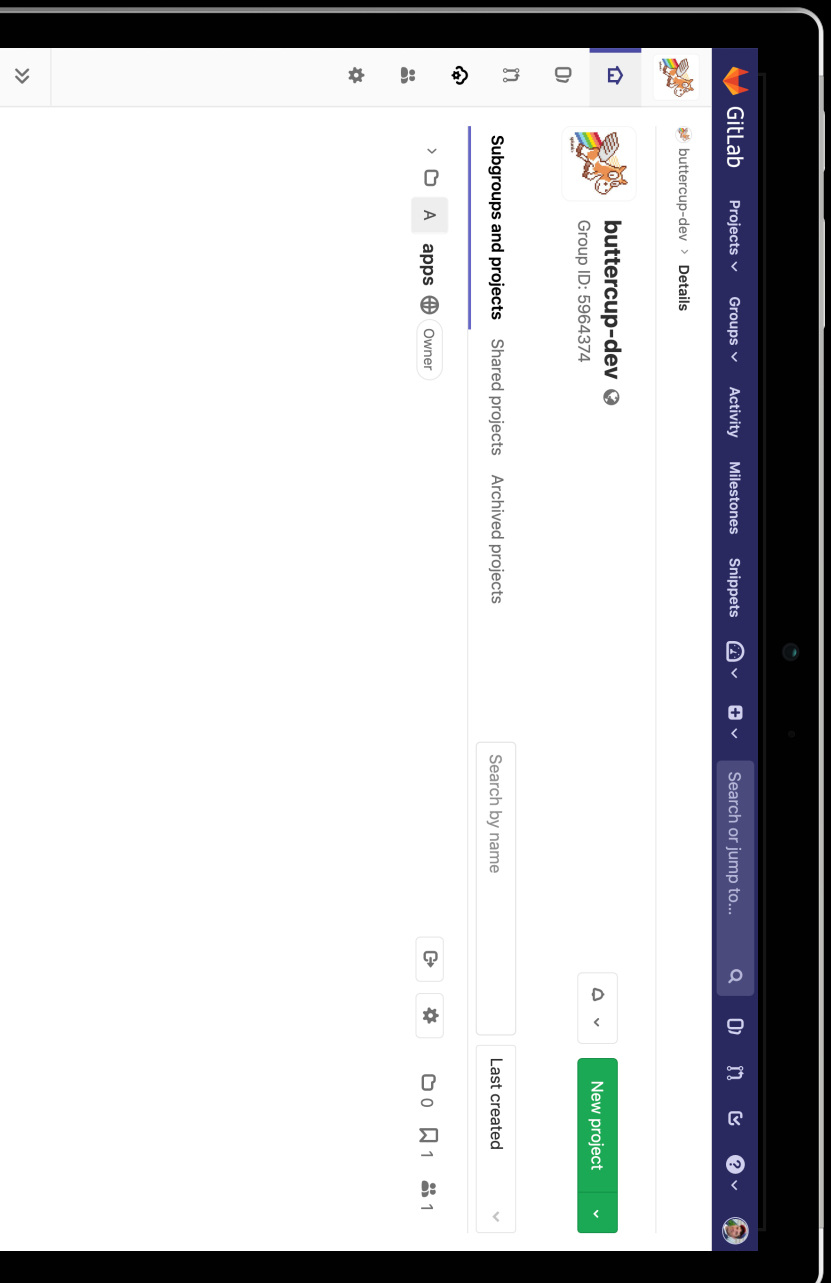
“combines the simplicity of a source code editor with powerful developer tooling”



- ▶ Available for macOS, Linux, and Windows
- ▶ Extensible
 - Hundreds of Extensions
 - Splunk Config File Editor
- ▶ Edit, build, and debug Splunk App Files with ease
- ▶ Integrated Terminal

Getting Started with GitLab

“Single Application For Entire Software Development And Operations Lifecycle”



- ▶ Source Control For Splunk App Files
- ▶ Version Control
- ▶ Easy Integration
- ▶ Free

GitLab

Structure Overview

<https://gitlab.com/Group/Subgroup/Project>

https://gitlab.com/buttercup-dev/apps/buttercup_app_for_splunk

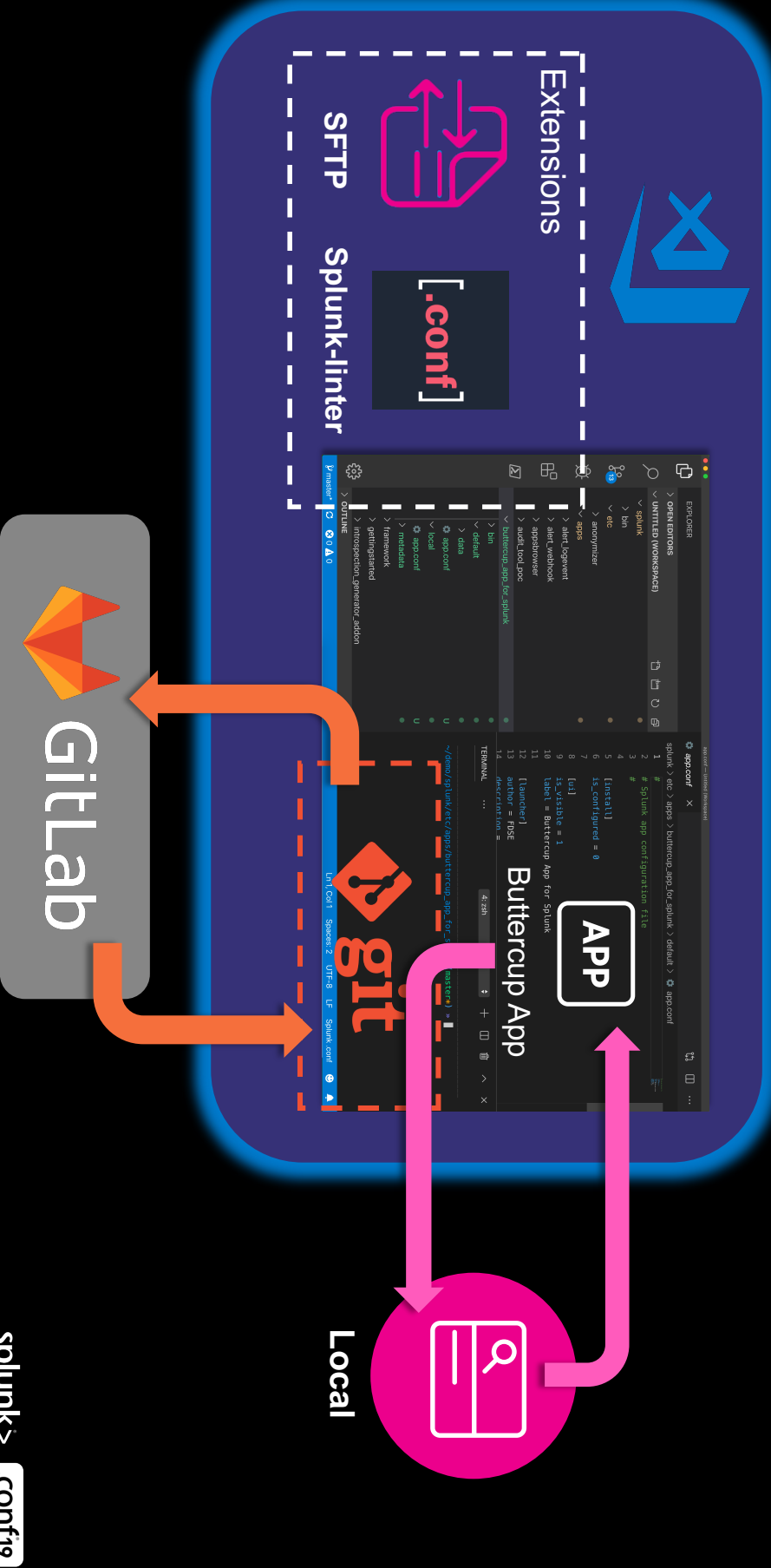
- ▶ **Group(s)** = team(s): Could be internal team -> e.g. **buttercup-dev**
- ▶ **Subgroup(s)** = group of projects: Could be Splunk Apps -> e.g. **apps**
- ▶ **Project(s)** = repository(s): individual Splunk app -> **buttercup_app_for_splunk**

* This is a public repository, seriously! Check it out after this presentation.

* Creation of a Subgroup is **NOT** mandatory

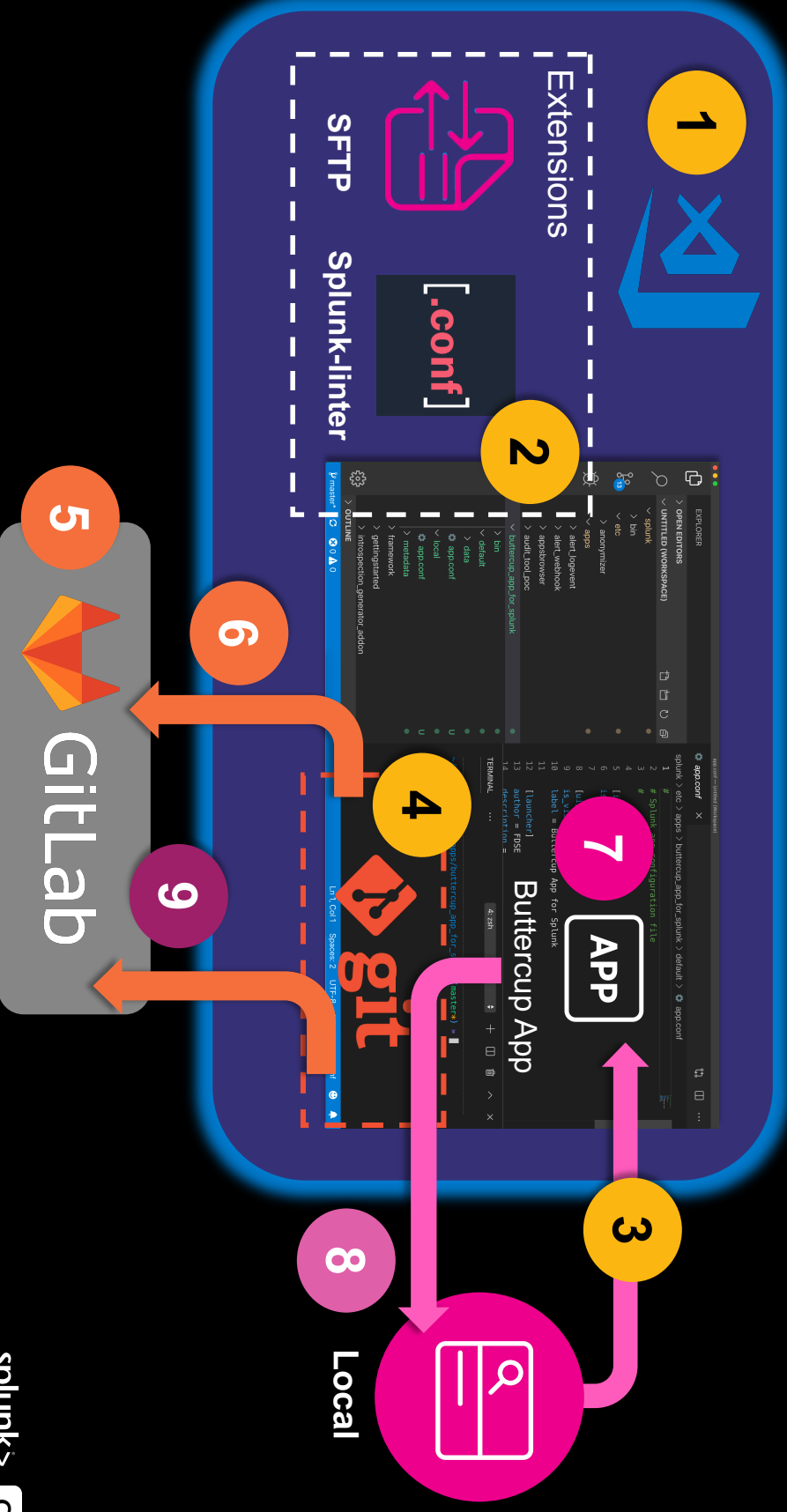
How They Work Together

Architecture



How They Work Together

Architecture

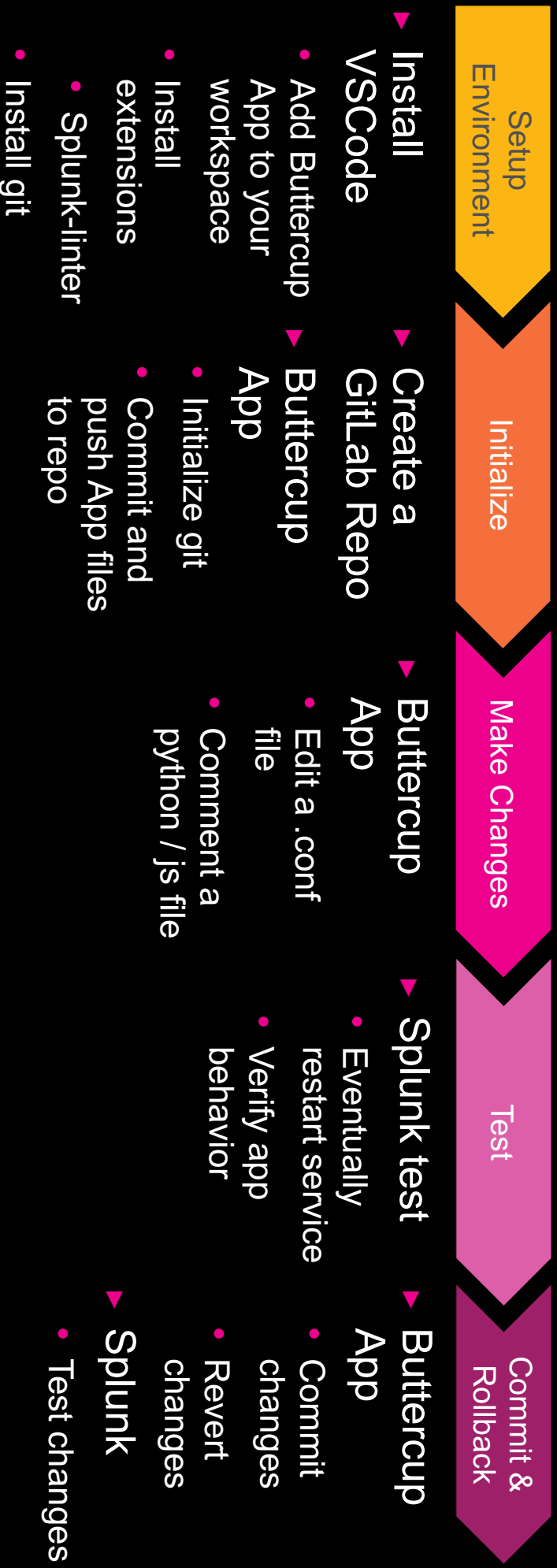




Local

Use Case 1

Required Steps To Take With Local Splunk Instance



splunk> **.conf19**



Use Case 2

Remote
Required Steps To Take With Remote Splunk Servers
Same as Use Case 1 **With Following Exceptions**

Setup Environment

▶ Install VSCode

- Install extensions

- Splunk-linter

• SFTP

- **Configure SFTP**

▶ **Test SSH connection**

Initialize

▶ Create a GitLab Repo

▶ Buttercup App

- **Sync Remote > Local**

- Initialize git

- Commit and push App files to repo

Make Changes

▶ Buttercup App

- Edit a .conf file

- Comment a python / js file

- **Sync Local > Remote**

Test Remote

▶ Splunk test

- Eventually restart service
- Verify app behavior

Commit & Rollback

▶ Buttercup App

- Commit changes

- Revert changes

- **Sync Local > Remote**

▶ Splunk

- Test changes

splunk> `conf`



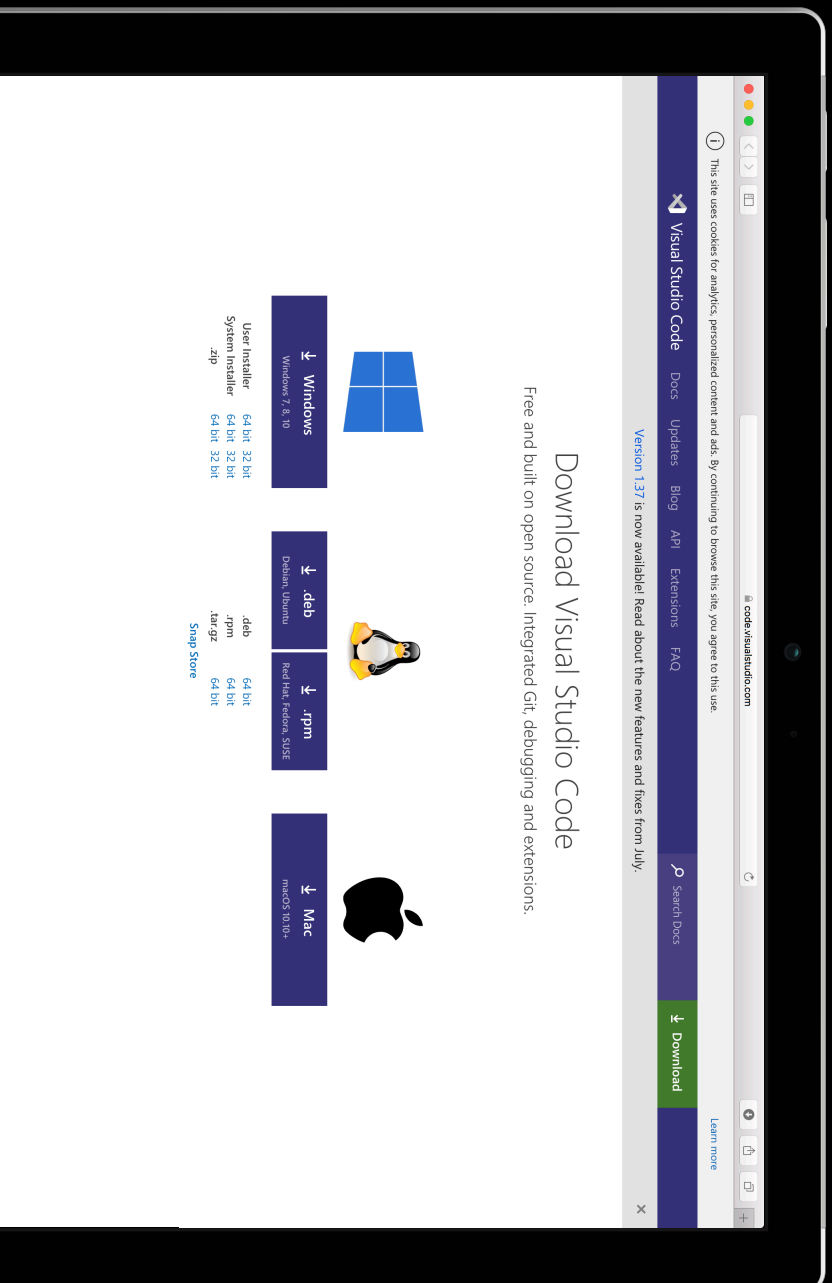
Demo

Use Case 1



Setup Environment

Install VSCode



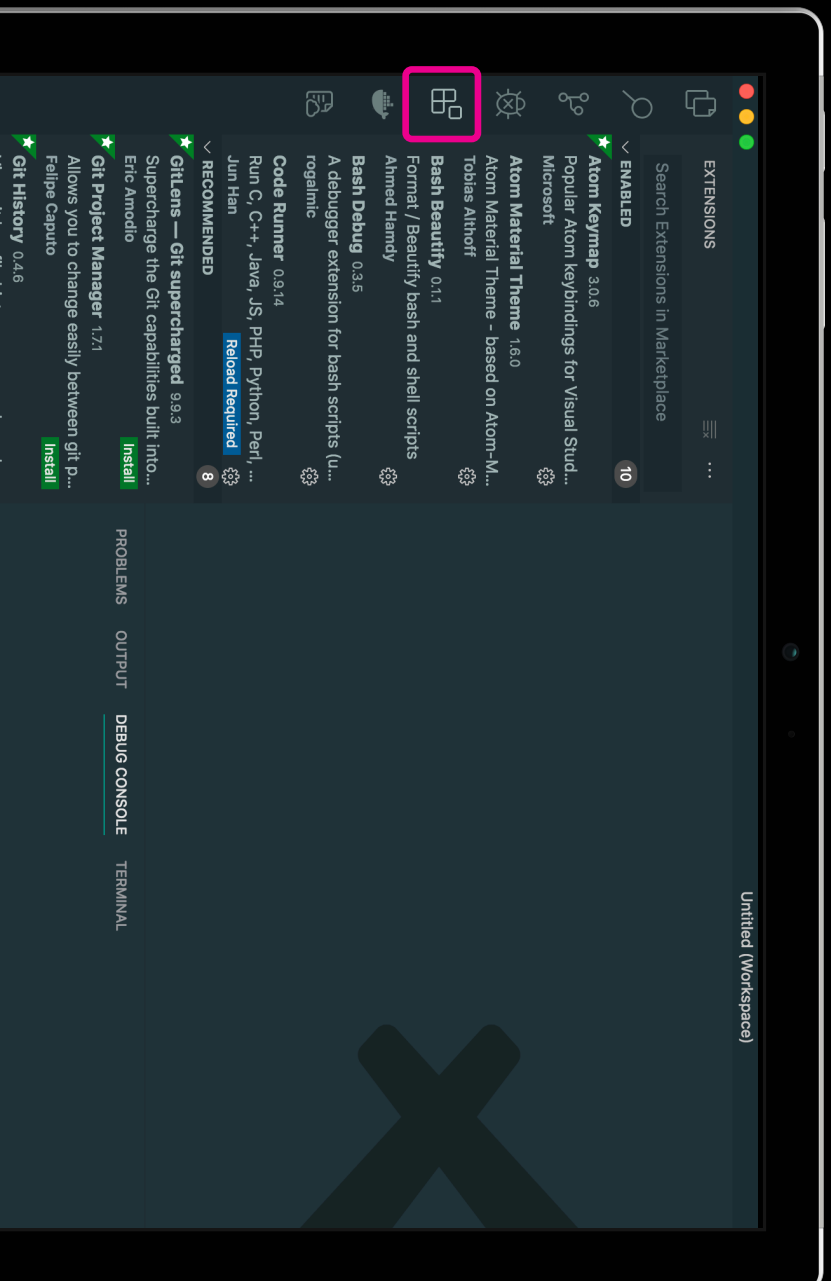
▶ Download VSCode in your local environment

- <https://code.visualstudio.com/Download>

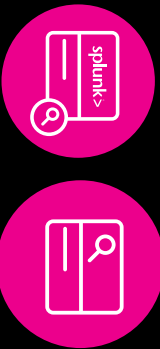


Setup Environment

Install VSCode Extensions

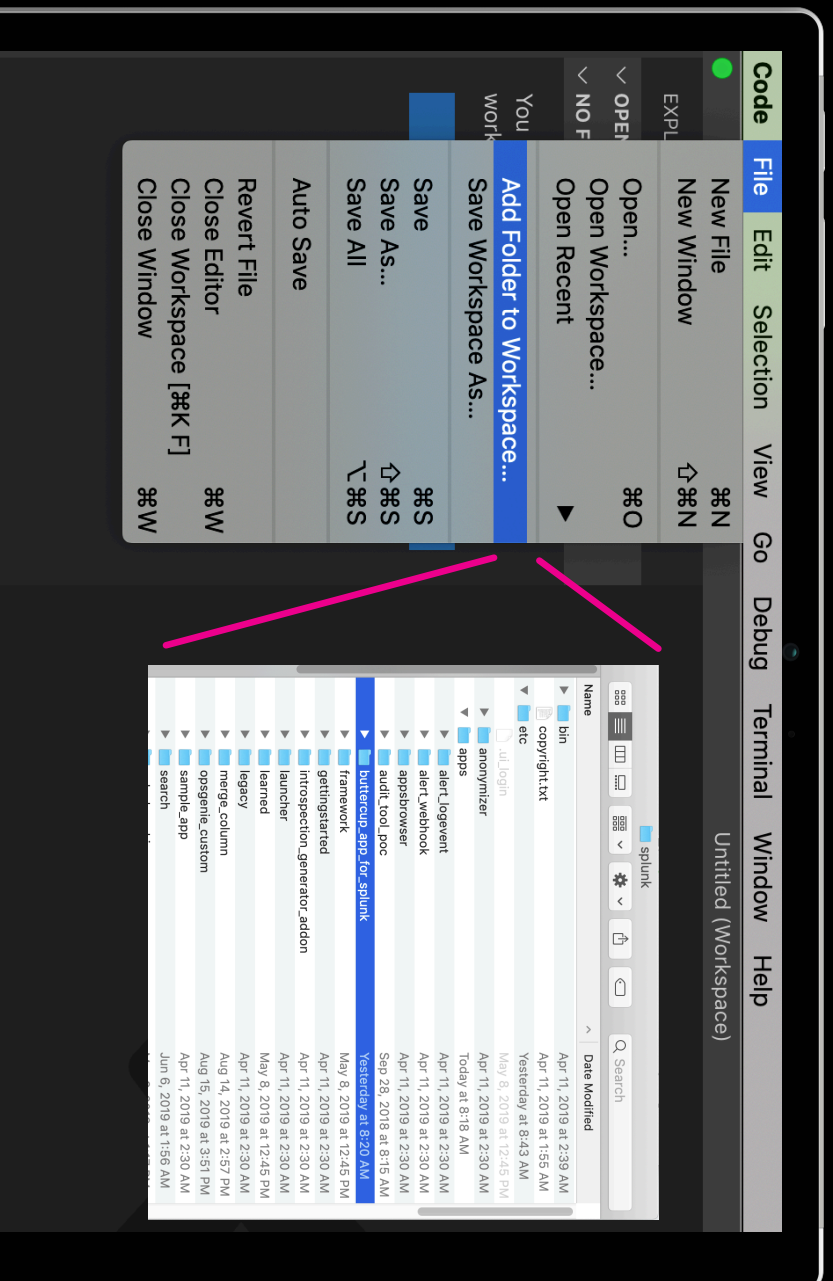


- ▶ Type your extension name in the search bar and select the desired one below
- ▶ Follow instructions displayed for
 - [Splunk-linter](#)
 - [SFTP](#)
 - Local-remote data synchronization
 - One configuration file per project / app



Setup Environment

Add Splunk App Folder to Workspace

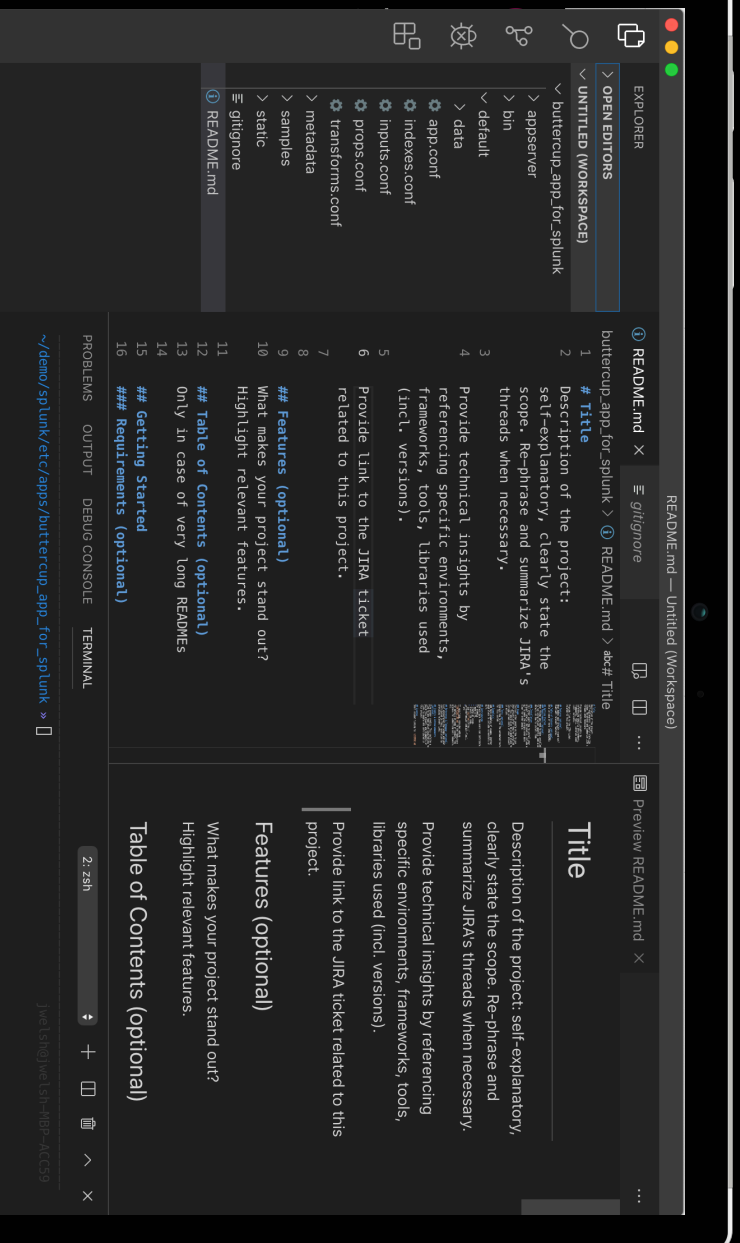


- ▶ Go to File -> Add Folder to Workspace
- ▶ Select buttercup_app_for_splunk from your Splunk apps directory



Initialize

Essential Files To Your Splunk Application

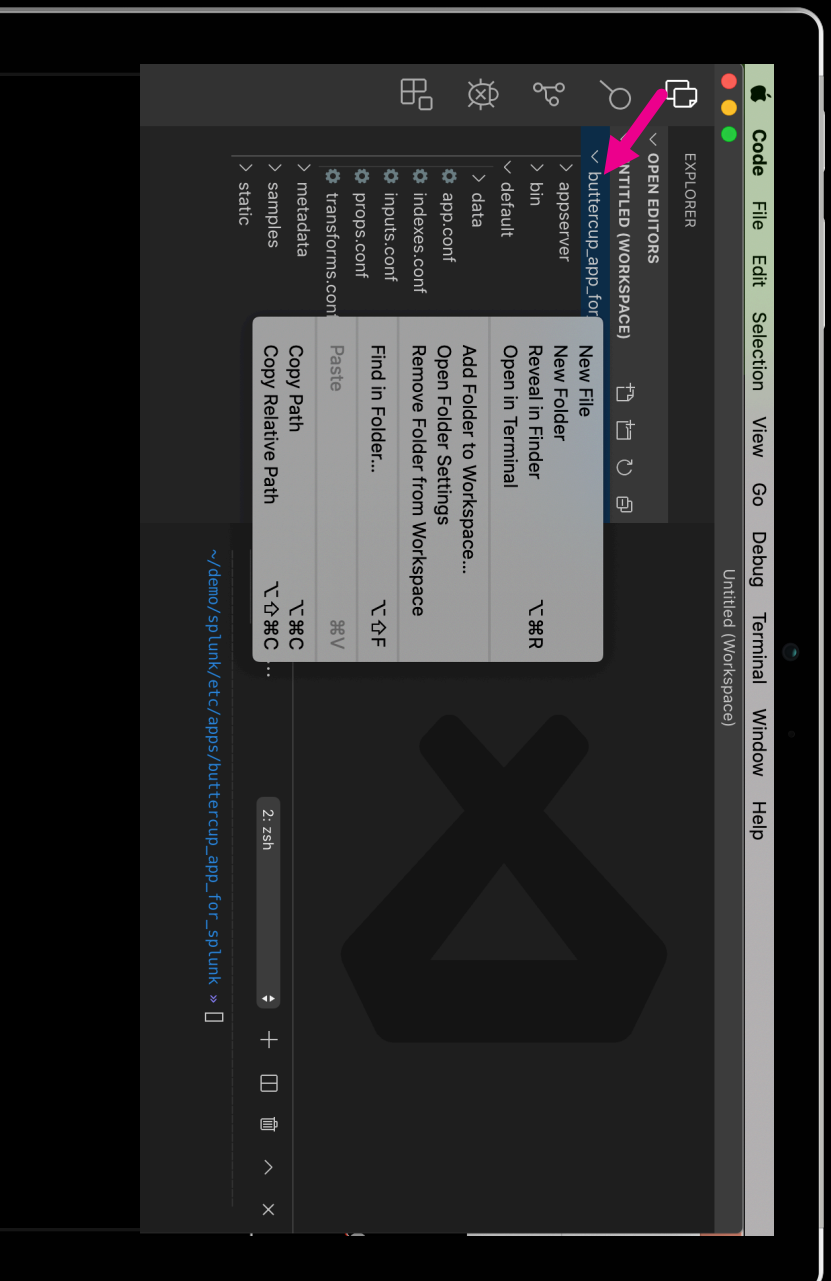


- ▶ Add README file
- Set expectations and manage contributions
- Give usage instructions
- ▶ Add .gitignore file
- Tells Git to ignore hidden and / or OS_ files



Setup Environment

Open Terminal Relative to buttercup_app_for_Splunk

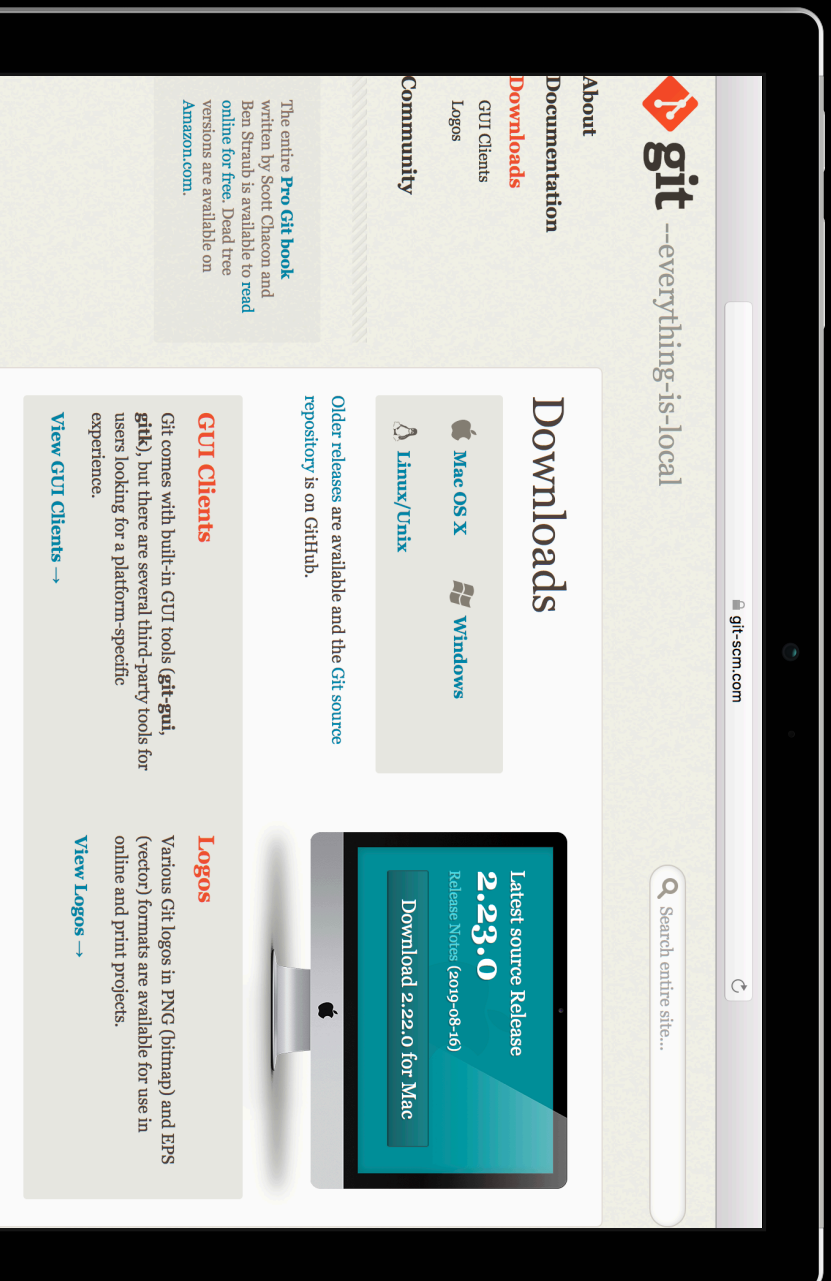


- ▶ Right click on the app directory
- ▶ Click on Reveal in Terminal



Setup Environment

Prepare Your Local System For Git



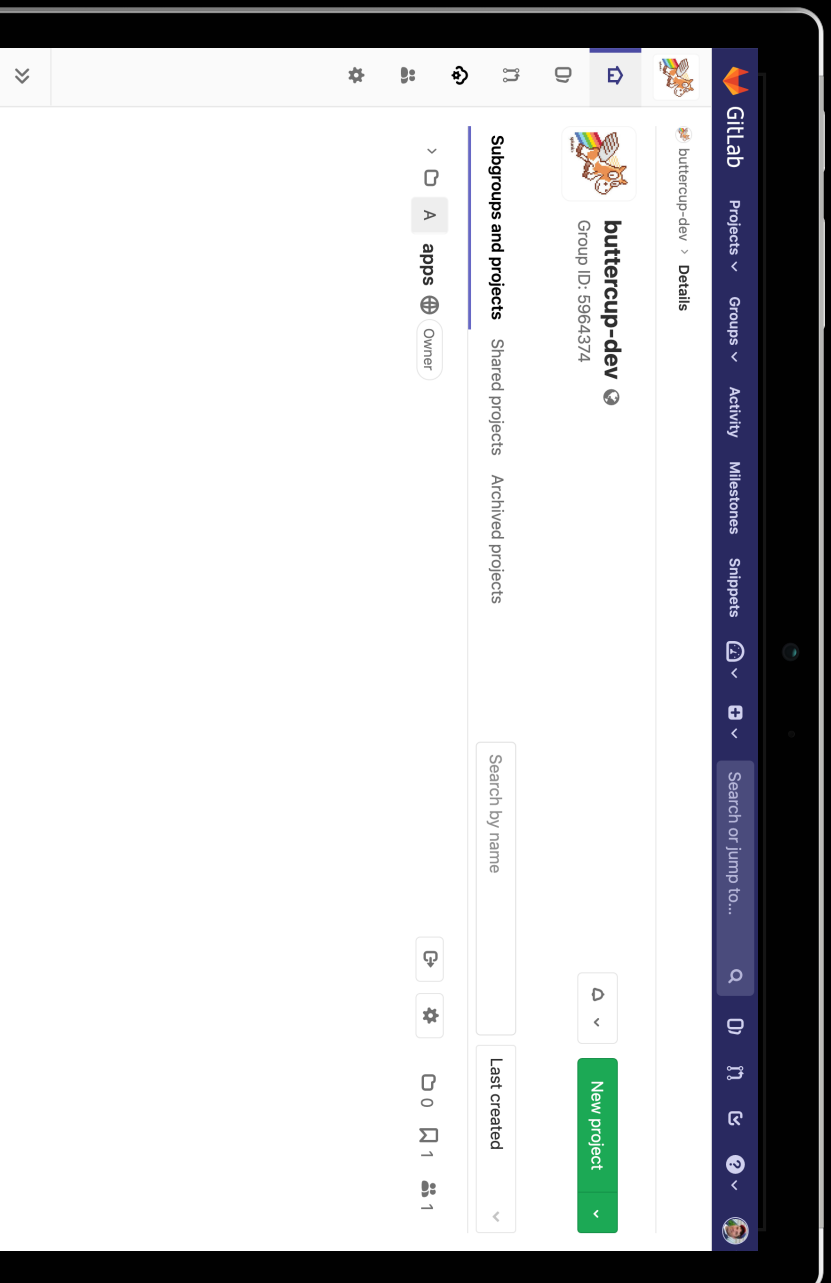
- ▶ Download Git in your local environment
 - <https://git-scm.com/downloads>
- ▶ Install Git
- ▶ Verify Git Works
 - `git --version`



Initialize

Getting Started with GitLab

A Free Source Code Repository with Version Control For Your Splunk App Files

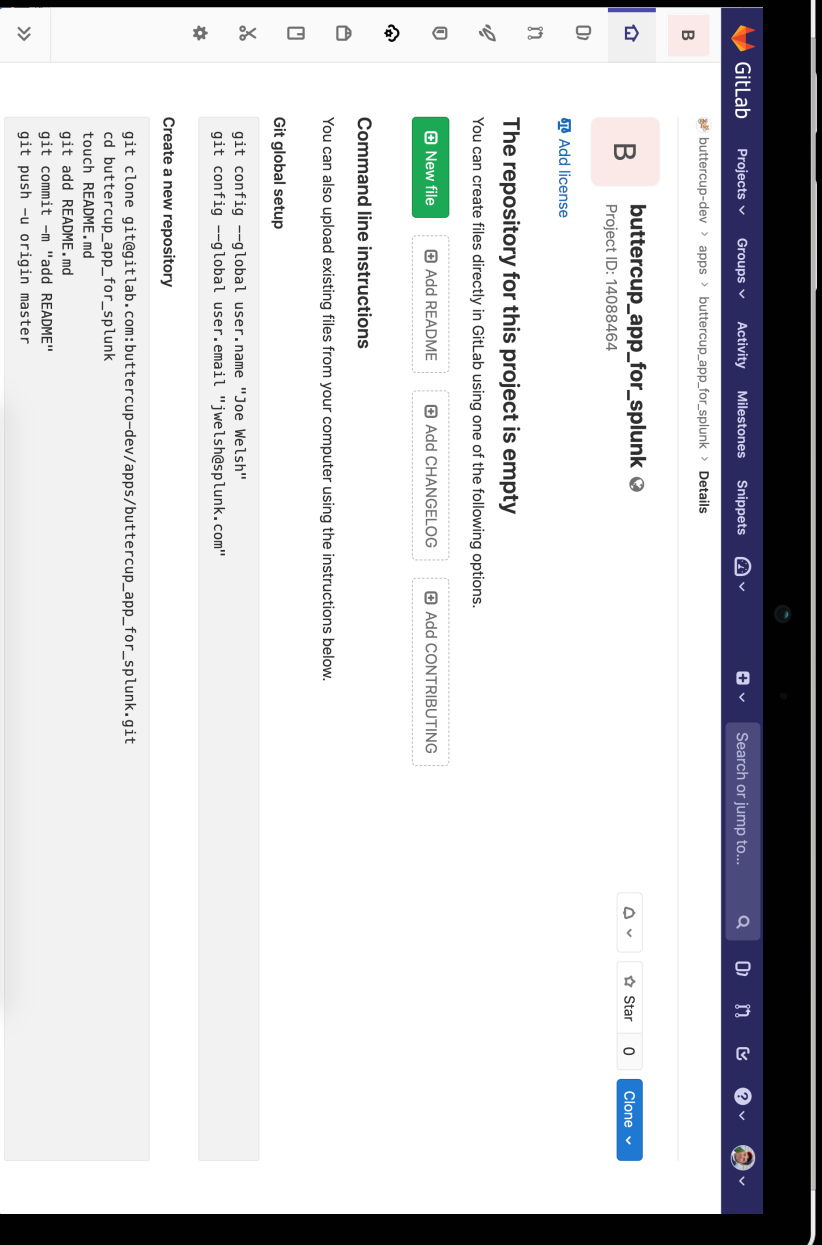


1. [Sign-in Or Register](#)
2. Choose “Free” Plan
3. [Create A Group](#)
4. [Add Users To The Group](#)
5. [Create Subgroups](#)
6. [Create Project In Subgroup](#)



Initialize

Create Project/Repository for buttermcup_app_for_splunk



1. Create Project In Subgroup

2. Copy or Bookmark instructions for “Git global setup” & “Push an existing folder” on command line inside Splunk app directory on your system.



Initialize

Commit Splunk App to Git

```
$~ cd $SPLUNK_HOME/etc/apps/buttercup_app_for_splunk  
$~ git init
```

1

Create an empty git repository locally

```
$~ git remote add origin git@gitlab.com:buttercup-dev/apps/buttercup_app_for_splunk.git
```

2

Add the remote repository to your current one

```
$~ git add .  
$~ git commit -m "Initial commit"
```

3

Add files to the repository and save local changes

```
$~ git push -u origin master
```

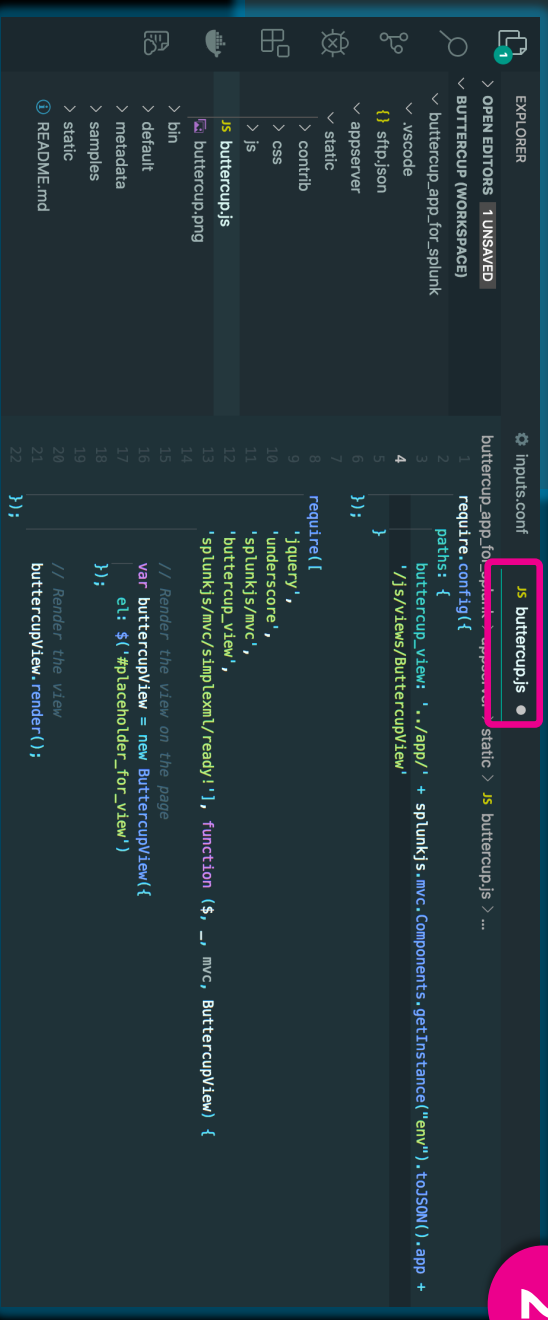
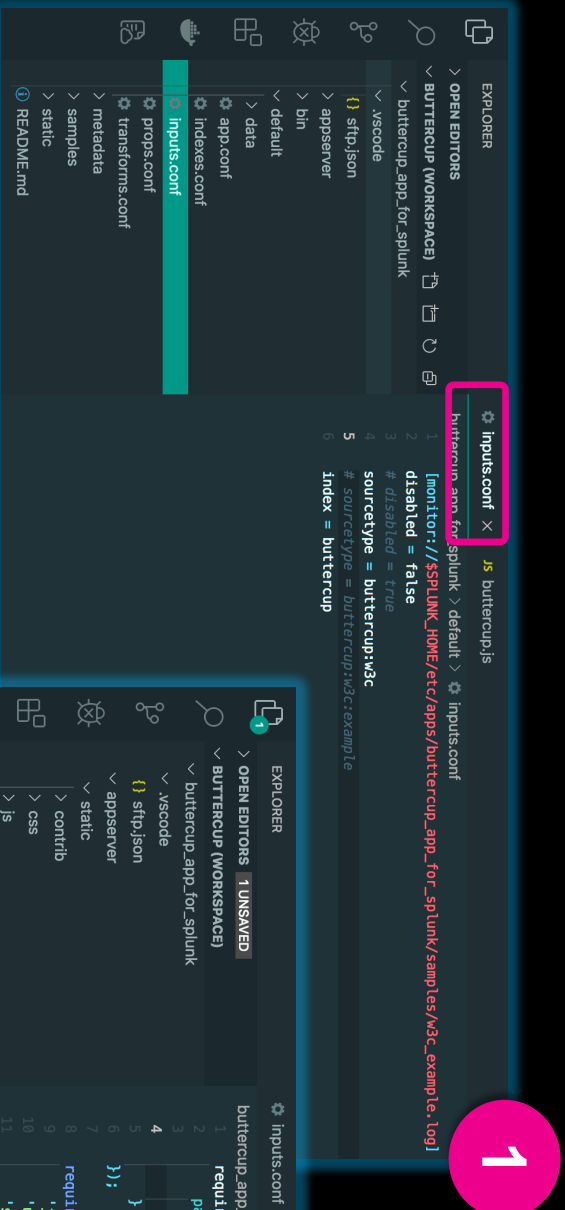
4

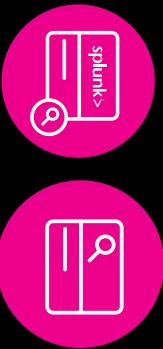
Push changes to server



Make Changes

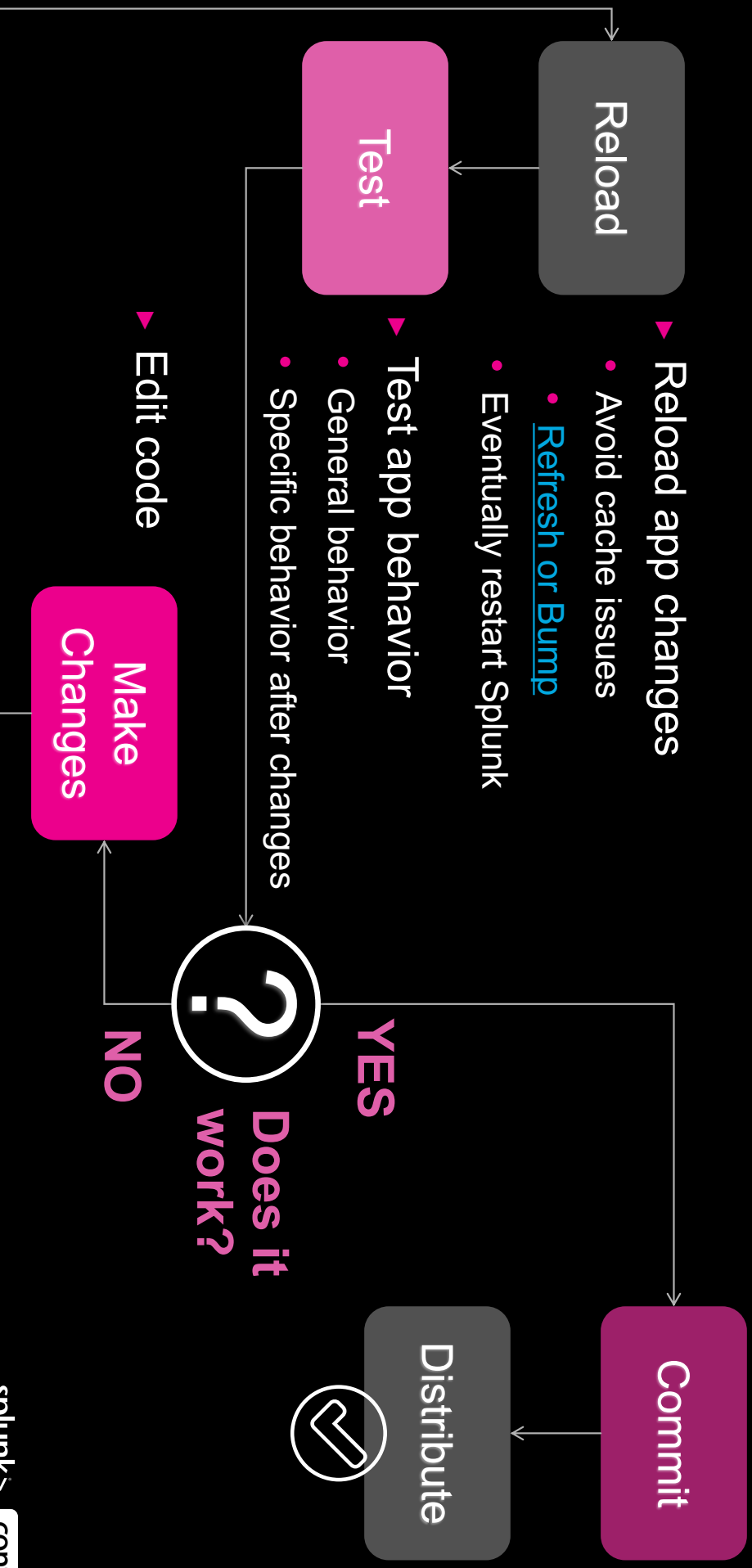
Edit inputs.conf and other source files





Test

Basic workflow for testing operations





Commit & Revert

Save changes to repository for version control

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

→ `buttercup_app_for_splunk git:(master) ✕ git commit -a -m "Updated inputs.conf"`

1

Save local changes to the repo

2

→ `buttercup_app_for_splunk git:(master) ✕ git push origin master`

-a (--all)

Automatically stage files that have been modified and deleted, but new files you have not told Git about are not affected.

-m <msg> (--message=<msg>)

Use the given <msg> as the commit message. If multiple -m options are given, their values are concatenated as separate paragraphs.

Commit Options
[More? [Click here](#)]

Push changes to server



Commit & Revert

Revert changes from previous version

You committed your code and then pushed it to the remote repository.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
➔ buttercup_app_for_splunk git:(master) ✕ git revert HEAD
```

Revert to previous version

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
➔ buttercup_app_for_splunk git:(master) ✕ git revert a3ad82c0
```

Revert to commit a3ad82c0

You haven't published your commits to a remote repository yet.

```
$ git stash
$ git reset --hard <hash-or-ref>
$ git stash pop
```

Revert to previous version saving local working changes



Demo

Use Case 2



Use Case 2

Remote
Required Steps To Take With Remote Splunk Servers
Same as Use Case 1 **With Following Exceptions**

Setup
Environment

▶ Install
VSCode

- Install extensions

- Splunk-linter

• **SFTP**

• **Configure SFTP**

▶ **Test SSH connection**

Initialize

▶ Create a
GitLab Repo

▶ Buttercup
App

• **Sync Remote > Local**

- Initialize git

- Commit and push App files to repo

Make Changes

▶ Buttercup
App

- Edit a .conf file

- Comment a python / js file

• **Sync Local > Remote**

Test Remote

▶ Splunk test

- Eventually restart service
- Verify app behavior

Commit &
Rollback

▶ Buttercup
App

- Commit changes

- Revert changes

• **Sync Local > Remote**

▶ Splunk

- Test changes

splunk> **.conf'r**



Setup Environment

Configure SFTP Extension for VSCode

1

Ctrl+Shift+P
cmd+shift+p

2

>SFTP: Config

3

`{ } sftp.json`

4

```
{
  "name": "Buttercup @buttercupRemotel",
  "host": "ec2-52-18-254-155.eu-west-1.compute.amazonaws.com",
  "protocol": "sftp",
  "port": 22,
  "privateKeyPath": "/Users/epescio/Downloads/erica-k.pem",
  "username": "ec2-user",
  "remotePath": "/opt/splunk/etc/apps/buttercup_app",
  "uploadOnSave": true,
  "ignore": [
    ".vscode",
    ".git",
    ".DS_Store"
  ]
}
```

- ▶ Provide host, username, protocol, etc
- ▶ Careful at:
 - Corresponding local and remote files structure
 - Permissions of remote folder

splunk>

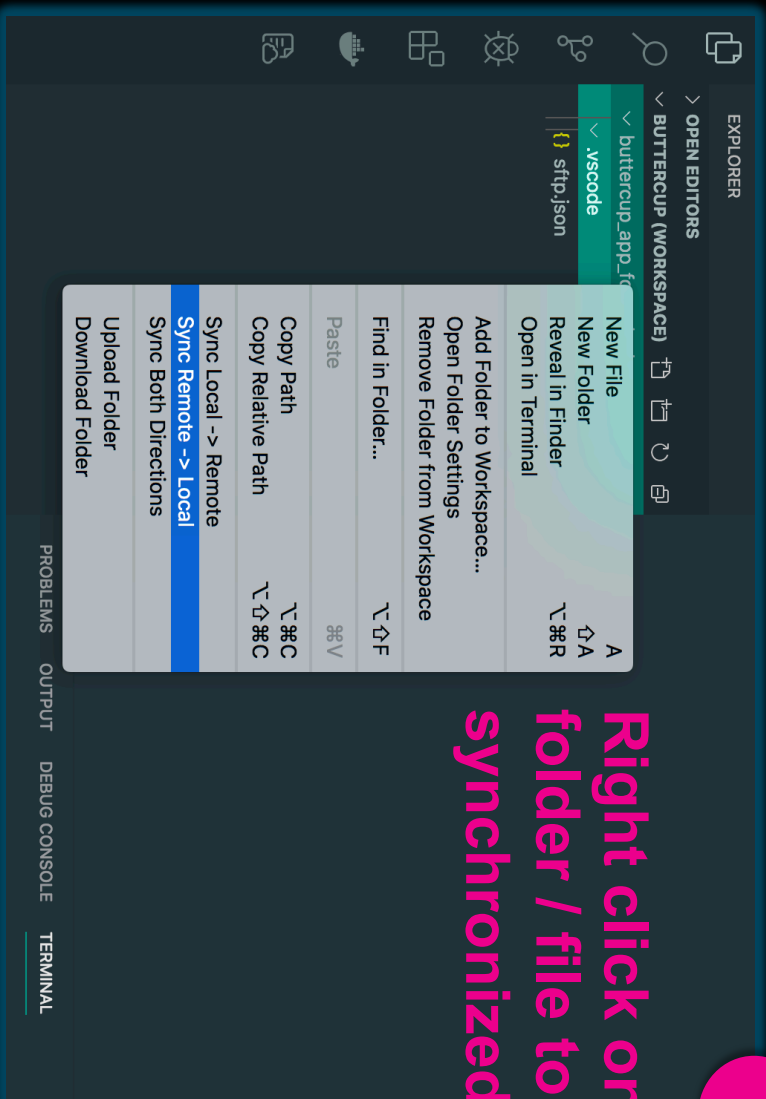
conf19



Make Changes

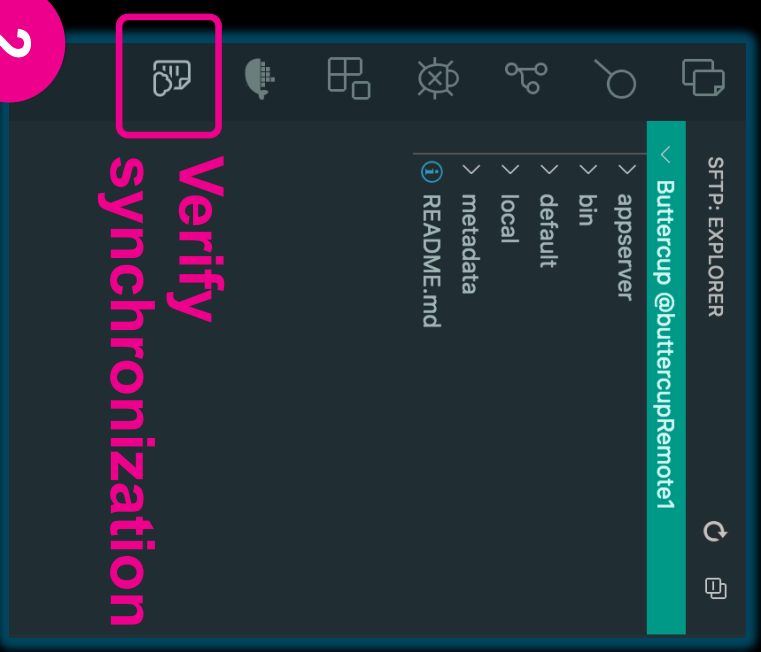
Synchronize app remote > local

1



Right click on folder / file to be synchronized

2



Verify synchronization



Make Changes

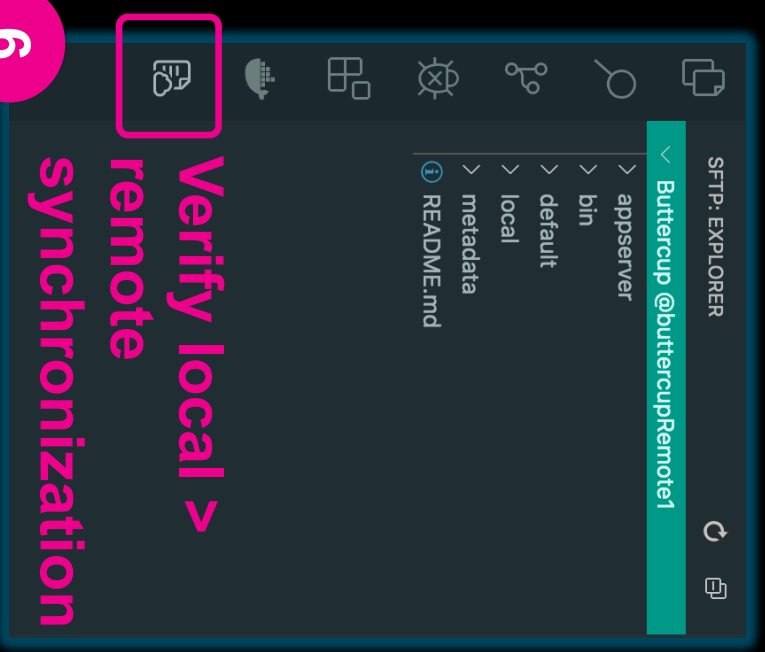
Synchronize app local > remote

5



Right click on folder / file to be synchronized

6

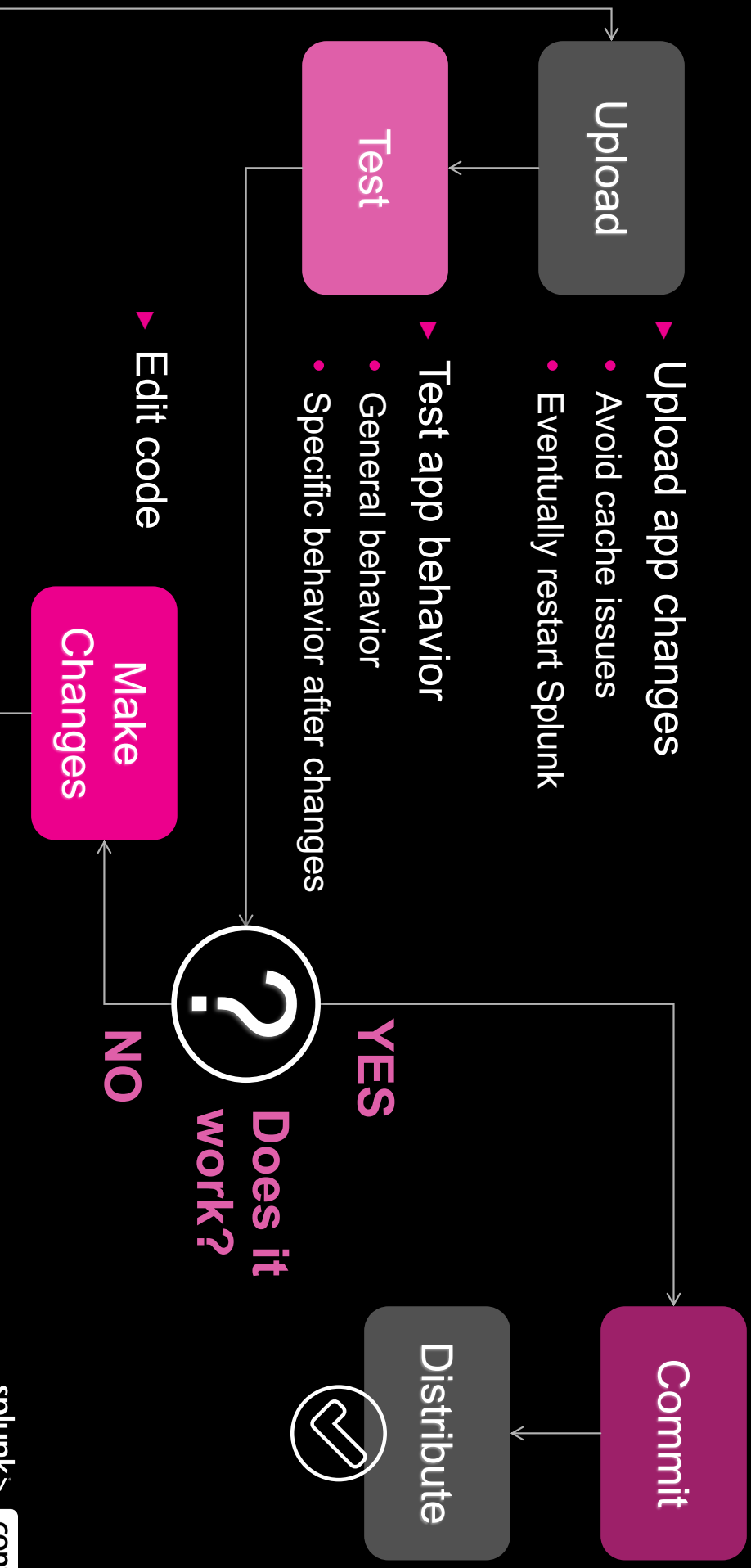


Verify local > remote synchronization



Test

Basic workflow for testing operations





Benefit

What's The Reason We Want To Do This, Again??



Recap – What's the Benefit to You

Integrating

- ▶ A Psuedo-Splunk Config Editor
 - Splunk Code Completion and Syntax Highlighting
- ▶ Less Errors, Reduced Development Cycles
 - Easily Revert Changes
- ▶ Version Control and Application Backup
- ▶ Simple synchronizing for editing remote files
- ▶ A single interface to do it all!

What You Should Do Next

Go get your hands wet! Try it out!!!



Q&A

Erica Pescio | Sr. FDSE Engineer

Joe Welsh | Sr. Manager, FDSE





**Thank
You!**