LEARNING ember-cli

#ember-cli

Table of Contents

About

Chapter 1: Getting started with ember-cli

Remarks

This section provides an overview of what ember-cli is, and why a developer might want to use it.

It should also mention any large subjects within ember-cli, and link out to the related topics. Since the Documentation for ember-cli is new, you may need to create initial versions of those related topics.

Simple syntax to create a project is :

```
ember new my-new-app
cd my-new-app
ember s
```

Kindly check instruction to setup ember-cli in this document

Examples

Installation

Ember-cli first requires Node and NPM to be installed on the system. Either follow the installation instructions on nodejs.org, or use a preferred package manager (such as Homebrew on OSX.) It's recommended to install latest version of each.

Once its done, run the following commands to ensure installation was correct:

```
node -v
npm -v
```

Since Yarn package manager has been released recently (October 2016), it's possible to install dependencies with Yarn instead of NPM. Checking the guide on yarn's website for further details.

Next, install Ember CLI globally:

```
npm install -g ember-cli
```

OR

```
yarn global add ember-cli
```

This will grant access to the ember command-line runner.

BOWER

Globally install Bower, a package manager that keeps front-end dependencies up-to-date. (including jQuery, Ember, and QUnit)

```
npm install -g bower
```

OR

```
yarn global add bower
```

This will grant access to the bower command-line runner.

PhantomJS

With Ember CLI, use a preferred automated test runner. Most testing services recommend or require PhantomJS, which can be installed via npm or the PhantomJS website. (PhantomJS is the default test runner for Testem and Karma.)

To use PhantomJS for integration tests, it must be globally installed:

```
npm install -g phantomjs-prebuilt
```

or

```
yarn global add phantomjs-prebuilt
```

Watchman

On OSX and UNIX-like operating systems, it is recommended to install Watchman version 4.x. This provides Ember CLI a more effective way for watching project changes.

File-watching on OSX is error-prone and Node's built-in `NodeWatcher` has trouble observing large trees. Watchman solves these problems and performs well on extremely massive file trees.

On OSX, install Watchman using Homebrew:

```
brew install watchman
```

For complete installation instructions, refer to the docs on the Watchman website.

Do *not* use an NPM version of Watchman. The following command can be used to uninstall it:

```
npm uninstall -g watchman
```

Congratulations! Now you are able to create your first project by running:

```
ember new my-first-app
```

start Ember server by running :

```
ember s
```

Navigate to `http://localhost:4200` to see the new app in action.

Navigate to `http://localhost:4200/tests` to see the test results in action.

Read Getting started with ember-cli online: https://riptutorial.com/ember-cli/topic/7441/getting-started-with-ember-cli

Chapter 2: Ember-cli Pods structure

Syntax

- Ember g [blueprints.eg: route] [name] --pod
- Ember g route foo --pod
- Ember g component my-name --pod

Parameters

Generate	pods
g	--pod

Remarks

Just pass `--pod` to `ember generate` when generating new files.

If you would like to use the pods structure as the default for your project, you can set usePods in your `.ember-cli` config file to true (setting was previously named `usePodsByDefault`). To generate or destroy a blueprint in the classic type structure while `usePods` is `true`, use the `--classic` flag.

With the usePods set to true.

```
// .ember-cli
{
    "usePods": true
}
```

The following would occur when generating a route:

```
ember generate route taco

installing
  create app/taco/route.js
  create app/taco/template.hbs
installing
  create tests/unit/taco/route-test.js

ember generate route taco --classic

installing
  create app/routes/taco.js
  create app/templates/taco.hbs
installing
  create tests/unit/routes/taco-test.js
```

There are some benefits to use this method, however, it's completely up to you.Firstly, it separates your application into more logical groupings, thus, you can keep your files neatly organized into resources.

This structure also makes our development's life easier. For instance, if I want to find the `myname controller` in the default structure, I need to preface what I actually want (myname) with the type (controllers). However, with pods, I can fuzzy-find the same controller by simply looking up “myname.”

Examples

Organize with Pods

```
app/controllers/myname.js
app/templates/myname.hbs
app/routes/myname.js
app/models/myname.js
```

Using pods, the example above would translate into this:

```
app/myname/controller.js
app/myname/template.hbs
app/myname/route.js
app/myname/model.js
```

podModulePrefix: app/pods

```
ember generate route foo --pod

installing
  create app/pods/foo/route.js
  create app/pods/foo/template.hbs
installing
  create tests/unit/pods/foo/route-test.js
```

Read Ember-cli Pods structure online: https://riptutorial.com/ember-cli/topic/7855/ember-cli-pods-structure

Chapter 3: Getting Started with Ember-Cli and Deployments

Syntax

- ember deploy production // deploy production environment
- ember deploy staging // deploy staging environment
- ember deploy development // deploy development environment which is not compress and minified

Parameters

parameters	details
`ember help`	show all possible params and depth guide as well as shortcodes

Remarks

Ember-Cli is a powerful tool which comes with many others to help us deploying faster and convenient. All you need to install Ember-Cli-Deploy and use `ember deploy`.

> Ember CLI Deploy structures your app's deployment using a deploy pipeline, which consists of several pipeline hooks. These standard hooks are the foundation for a rich ecosystem of plugins which you can compose to create a deployment process suitable for your application.

As Ember-Cli-Deploy is an addon of Ember so you can easily install that with `ember install ember-cli-deploy`. There are two useful other add-ons which make our build and compressing reliable during deployment.

Simply run the following commands:

```
# Install the Build plugin, which builds your app during deployment
ember install ember-cli-deploy-build

# Gzip our files
ember install ember-cli-deploy-gzip
```

However, if you are going to maximize your benefits using `ember deploy`, it's most likey to have different environments in your Ember application and deploy production,staging or development version of your app with the appropriate configuration.

Platforms that you can deploy by now are:

- Heroku
- Azure
- AWS S3
- Firebase
- CouchDB cluster

Kindly refer to example section to see how you can deploy.

Examples

Heroku

You must have installed Heroku Toolbelt first. Having an account in Heroku and installation of ember-cli-deploy are mandatory.

Creating a new Heroku instance from an Ember CLI application's parent directory:

```
$ heroku create --buildpack https://github.com/tonycoco/heroku-buildpack-ember-cli.git

$ git push heroku master
```

Azure

Firstly, it's required to install Microsoft’s module ember-cli-azure-deploy. You need to be in your application root directory.

```
npm install --save-dev -g ember-cli-azure-deploy
azure-deploy init
```

If you are using Yarn package manager you can simply install by:

```
yarn global add ember-cli-azure-deploy
azure-deploy init
```

This will create a `deploy.sh` in your project's root folder, enabling Azure to follow a set of instructions - including installing all the required Node Modules, running `ember build` and deploying the resulting `dist/` folder to your website's `wwwroot`.

Firebase

First, you need to install Firebase tools. Simply, run the commands below:

Npm Package manager

```
npm install -g firebase-tools
```

or Yarn package manager

```
yarn add firebase-tools
```

To configure your application to be ready to deploy you need to run the following in your app’s root directory:

```
firebase init
```

finally, by running the following command you can deploy your application

```
firebase deploy
```

AWS S3

To proceed with the deployment to S3, we will install these plugins:

- ember-cli-deploy-s3-index: It uploads the index.html to S3 with revision information, and activates it
- ember-cli-deploy-s3: It uploads the assets (js, css and other media files) to S3

As they are Ember addon you can easily install by running the following commands

```
ember install ember-cli-deploy-s3-index
ember install ember-cli-deploy-s3
```

All you need after that is to configure deploy.js file which should under /config folder:

```
// config/deploy.js

module.exports = function(deployTarget) {

  var ENV = {
    build: {
      environment: deployTarget
    },
    'revision-data': {
      type: 'git-commit'
    },
    's3-index': {
      accessKeyId: process.env['S3_ACCESS_KEY'],
      secretAccessKey: process.env['S3_SECRET_ACCESS_KEY'],
      bucket: "your-app-deployment-bucket",
      region: "YOUR REGISION",
      allowOverwrite: true // if you want to overwrite index file if not change it to false
    },
    's3': {
      accessKeyId: process.env['S3_ACCESS_KEY'],
      secretAccessKey: process.env['S3_SECRET_ACCESS_KEY'],
      bucket: "your-app-deployment-bucket",
      region: "YOUR REGISION",
    }
  };

  return ENV;
};
```

Notice that we have used the environmental variables within our config.If you need to read configuration from a file, it’s also possible to return a promise that resolves with the `ENV` object. Here is an example to define different environments in deploy.js file:

```
if (deployTarget === 'development') {
    ENV.build.environment = 'development';
    // configure other plugins for development deploy target here
  }

  if (deployTarget === 'staging') {
    ENV.build.environment = 'production';
    // configure other plugins for staging deploy target here
  }

  if (deployTarget === 'production') {
    ENV.build.environment = 'production';
    // configure other plugins for production deploy target here
  }
```

Finally, you can easily run following command to deploy

```
ember deploy [YOUR APP ENVIRONMENT] //e.g-> ember deploy production or ember deploy staging
```

if you like to see details you can run:

```
ember deploy production --verbose --activate=true
```

Read Getting Started with Ember-Cli and Deployments online: https://riptutorial.com/ember-cli/topic/7444/getting-started-with-ember-cli-and-deployments

Credits

S. No	Chapters	Contributors
1	Getting started with ember-cli	4444, Community, Majid
2	Ember-cli Pods structure	Majid
3	Getting Started with Ember-Cli and Deployments	Majid