**Step 1:** **Create a boilerplate app using react template**(Assuming you have already installed JSS Cli node module as described in pre-requisites above.)

Use the following commands to create the boilerplate app :

| **$ jss create jss-demo-app react** |
| --- |

To [create boilerplate app for a specific JSS version](https://github.comcast.com/sitecore-comcast/jss-demo-app/blob/master/README.md#to-create-the-react-app-for-a-specific-version-for-example-for-sitecore-93-recommended-jss-version-is-1300), e.g: 15.0.20 (recommended as per Sitecore documentation), use the following commands:

| **$ jss create jss-demo-app react -b release/15.0.0** |
| --- |

*Where, 'jss-demo-app' is the appName and 'react' is the template. Checkout*[*this*](https://jss.sitecore.com/docs/client-frameworks/getting-started/quick-start)*documentation for more info. This boilerplate app was bootstrapped with help of*[*https://github.com/facebook/create-react-app*](https://github.com/facebook/create-react-app)*.*

**Step 2: Start with disconnected mode**

Use the following commands to start the disconnected mode:

| **$ cd jss-demo-app**  **$ jss start**  **OR**  **$ npm run start** |
| --- |

Sitecore is not required to be installed, and content data is pulled from static local sources. Disconnected development is appropriate when no Sitecore instance is available, or when frontend developers do not wish to run a local copy of Sitecore. Content data is mocked using local files (json/yaml/js) instead of a Sitecore instance. You local app connects to mock sitecore proxy (you don't need to setup this, jss framework does it all for you) and delivers the mocked content data.

At this point you would cleanup your boilerplate app, add/remove templates/components etc. and test it locally in disconnected mode and make it ready for transition to connected mode.

**Step 3: Keep the Sitecore 'API key' handy, so that you can use it during config setup on the next step.**

**Step 4: Setup the jss configs (assuming you already have the API key for your app, check the prerequisites) for transition to connected mode.**

* Use the following commands to setup jss configs:

| **$ jss setup** |
| --- |

* Provide the details as asked on the prompt, keeping in mind that you need to connect to Sitecore 9.3 sandbox. At the end it would generate two files. Example:

| **Is your Sitecore instance on this machine or accessible via network share? [y/n]: y Path to the Sitecore folder (e.g. c:\inetpub\wwwroot\**[**my.siteco.re**](http://my.siteco.re/)**):  Sitecore hostname (e.g.**[**http://myapp.local.siteco.re**](http://myapp.local.siteco.re/)**; see /sitecore/config; ensure added to hosts):**[**https://sc93.local.com**](https://cms9.sbox.xfinity.com/) **Sitecore import service URL [**[**https://sc93.local.com/sitecore/api/jss/import**](https://cms9.sbox.xfinity.com/sitecore/api/jss/import)**]:**[**https://sc93.local.com/sitecore/api/jss/import**](https://cms9.sbox.xfinity.com/sitecore/api/jss/import) **Sitecore API Key (ID of API key item): {A0F9FC32-EBE5-46A4-A556-1FAB7F35DCEA} Please enter your deployment secret (32+ random chars; or press enter to generate one): Deployment secret has been generated. Ensure the JSS app config on the Sitecore end has the same secret set. Deploy secret Sitecore config written to C:\sitecoreFrontend\jss-demo-app\sitecore\config\jss-demo-app.deploysecret.config Ensure this configuration is deployed to Sitecore. JSS connection settings saved to C:\sitecoreFrontend\jss-demo-app\scjssconfig.json**  **NEXT STEPS \* Ensure the hostName in /sitecore/config/\*.config is configured as**[**sc93.local.com**](http://cms9.sbox.xfinity.com/)**, and in hosts file if needed. \* Deploy your configuration (i.e. 'jss deploy config') \* Deploy your app (i.e. 'jss deploy app -c -d') \* Test your app in integrated mode by visiting**[**https://sc93.local.com**](https://cms9.sbox.xfinity.com/) |
| --- |

* **"*scjssconfig.json*"** at the root of the project/app with content based on inputs provided. Example config:

| **{**  **"sitecore": {**  **"instancePath": "",**  **"apiKey": "{A0F9FC32-EBE5-46A4-A556-1FAB7F35DCEA}",**  **"deploySecret": "8zd8ca6rfdm7mygupa3pg6yq7buvcfdkceuz6x9g43f4",**  **"deployUrl": "**[**https://sc93.local.com/sitecore/api/jss/import**](https://cms9.sbox.xfinity.com/sitecore/api/jss/import)**",**  **"layoutServiceHost": "**[**https://sc93.local.com**](https://cms9.sbox.xfinity.com/)**"**  **}**  **}** |
| --- |

Note: Above config is just a sample and not for copying for your app. Generate the config for your app and compare with this example. It is just for reference.

* **"<appName>.deploysecret.config"** (in this case ***jss-demo-app.deploysecret.config***) at **"/sitecore/config"** path in the project/app with content based on inputs provided. Example config:

| **<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/">**  **<sitecore>**  **<javaScriptServices>**  **<apps>**  **<app name="jss-demo-app"**  **deploymentSecret="8zd8ca6rfdm7mygupa3pg6yq7buvcfdkceuz6x9g43f4" debugSecurity="false" />**  **</apps>**  **</javaScriptServices>**  **</sitecore>**  **</configuration>** |
| --- |

Note: Above config is just a sample and not for copying for your app. Generate the config for your app and compare with this example. It is just for reference.

* When you create the boilerplate app, it by default generates another config **"*<appName>.config*"** (in this case ***jss-demo-app.config***) under ***"/sitecore/config"****.*Make sure you update the hostName (***hostName="***[***sc93.local.com***](http://cms9.sbox.xfinity.com/)***"***) to match the sandbox hostname under site definition settings. Example config:

| **<!--**  **JSS Sitecore Configuration Patch File**  **This configuration file registers the JSS site with Sitecore, and configures the Layout Service**  **to work with it. Config patches need to be deployed to the Sitecore server.**  **Normally `jss deploy config` can do this for local development. To manually deploy, or to deploy via CI,**  **this file can be placed in the `App\_Config/Include` folder, or a subfolder of it, within the Sitecore site.**  **-->**  **<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/" xmlns:set="http://www.sitecore.net/xmlconfig/set/" xmlns:role="http://www.sitecore.net/xmlconfig/role/">**  **<sitecore>**  **<settings>**  **<!--**  **PROXY CONFIGURATION**  **When running JSS with the headless proxy (node-headless-ssr-proxy),**  **the original IP address of clients is obscured by the proxy.**  **This setting tells Sitecore to read the forwarded header the proxy sends,**  **thus making analytics track the correct original client IP address.**  **-->**  **<setting name="Analytics.ForwardedRequestHttpHeader" set:value="X-Forwarded-For" />**  **</settings>**  **<sites>**  **<!--**  **JSS Site Registration**  **This configures the site with Sitecore - i.e. host headers, item paths.**  **If your JSS app lives within an existing Sitecore site, this may not be necessary.**  **IMPORTANT: JSS sites ship in 'live mode', which makes development and testing easy,**  **but disables workflow and publishing. Before going to production, change the `database`**  **below to `web` instead of `master`.**  **-->**  **<site patch:before="site[@name='website']"**  **inherits="website"**  **name="jss-demo-app"**  **hostName="**[**sc93.local.com**](http://cms9.sbox.xfinity.com/)**"**  **rootPath="/sitecore/content/jss-demo-app"**  **startItem="/home"**  **database="master" />**  **</sites>**  **<javaScriptServices>**  **<apps>**  **<!--**  **JSS App Registration**  **The JSS app needs to be registered in order to support layout service and import services.**  **There are many available attributes, and they inherit the defaults if not explicitly specified here.**  **Defaults are defined in `/App\_Config/Sitecore/JavaScriptServices/Sitecore.JavaScriptServices.Apps.config`**  **NOTE: graphQLEndpoint enables \_Integrated GraphQL\_. If not using integrated GraphQL, it can be removed.**  **-->**  **<app name="jss-demo-app"**  **sitecorePath="/sitecore/content/jss-demo-app"**  **useLanguageSpecificLayout="true"**  **graphQLEndpoint="/api/jss-demo-app"**  **inherits="defaults"**  **/>**  **</apps>**  **<!--**  **IMAGE RESIZING WHITELIST**  **Using Sitecore server-side media resizing (i.e. the `imageParams` or `srcSet` props on the `<Image/>` helper component)**  **could expose your Sitecore server to a denial-of-service attack by rescaling an image with many arbitrary dimensions.**  **In JSS resizing param sets that are unknown are rejected by a whitelist.**  **Sets of image sizing parameters that are used in app components must be whitelisted here.**  **If a param set is not whitelisted, the image will be returned \_without resizing\_.**  **To determine the image parameters being used, look at the query string on the `src` of the rendered image, i.e. '/img.jpg?mw=100&h=72' -> mw=100,h=72**  **Note: the parameter sets defined here are comma-delimited (,) instead of &-delimited like the query string. Multiple sets are endline-delimited.**  **-->**  **<allowedMediaParams>**  **<!-- XML element name is arbitary, useful for organizing and patching -->**  **<styleguide-image-sample>**  **mw=100,mh=50**  **</styleguide-image-sample>**  **<styleguide-image-sample-adaptive>**  **mw=300**  **mw=100**  **</styleguide-image-sample-adaptive>**  **</allowedMediaParams>**  **</javaScriptServices>**  **<api>**  **<!--**  **Define the app's Sitecore GraphQL API endpoint**  **Note: this can be removed if you are not using GraphQL.**  **Note: the endpoint must be defined both for integrated and connected type GraphQL queries.**  **-->**  **<GraphQL>**  **<endpoints>**  **<jss-demo-appGraphQLEndpoint url="/api/jss-demo-app" type="Sitecore.Services.GraphQL.Hosting.DatabaseAwareGraphQLEndpoint, Sitecore.Services.GraphQL.NetFxHost" resolve="true">**  **<url>$(url)</url>**  **<enabled>true</enabled>**  **<enableSubscriptions>true</enableSubscriptions>**  **<!-- lock down the endpoint when deployed to content delivery -->**  **<graphiql role:require="ContentDelivery">false</graphiql>**  **<enableSchemaExport role:require="ContentDelivery">false</enableSchemaExport>**  **<enableStats role:require="ContentDelivery">false</enableStats>**  **<enableCacheStats role:require="ContentDelivery">false</enableCacheStats>**  **<disableIntrospection role:require="ContentDelivery">true</disableIntrospection>**  **<schema hint="list:AddSchemaProvider">**  **<content type="Sitecore.Services.GraphQL.Content.ContentSchemaProvider, Sitecore.Services.GraphQL.Content">**  **<!-- scope typed template generation to just this app's templates -->**  **<templates type="Sitecore.Services.GraphQL.Content.TemplateGeneration.Filters.StandardTemplatePredicate, Sitecore.Services.GraphQL.Content">**  **<database>context</database>**  **<paths hint="list:AddIncludedPath">**  **<templates>/sitecore/templates/Project/jss-demo-app</templates>**  **</paths>**  **<fieldFilter type="Sitecore.Services.GraphQL.Content.TemplateGeneration.Filters.StandardFieldFilter, Sitecore.Services.GraphQL.Content">**  **<exclusions hint="raw:AddFilter">**  **<!--**  **Remove system fields from the API (e.g. \_\_Layout) to keep the schema lean**  **-->**  **<exclude name="\_\_\*" />**  **</exclusions>**  **</fieldFilter>**  **</templates>**  **<queries hint="raw:AddQuery">**  **<!-- enable querying on items via this API -->**  **<query name="item" type="Sitecore.Services.GraphQL.Content.Queries.ItemQuery, Sitecore.Services.GraphQL.Content" />**  **</queries>**  **<fieldTypeMapping ref="/sitecore/api/GraphQL/defaults/content/fieldTypeMappings/standardTypeMapping" />**  **</content>**  **</schema>**  **<!-- Enables the 'jss' graph nodes that are preformatted to use with JSS rendering components, and the datasource resolving queries for JSS -->**  **<extenders hint="list:AddExtender">**  **<layoutExtender type="Sitecore.JavaScriptServices.GraphQL.JssExtender, Sitecore.JavaScriptServices.GraphQL" resolve="true" />**  **</extenders>**  **<!-- Determines the security of the service. 'publicService' is open to anonymous access, but requires an SSC API key. -->**  **<security ref="/sitecore/api/GraphQL/defaults/security/publicService" />**  **<!-- Determines how performance is logged for the service. Defaults are defined in Sitecore.Services.GraphQL.config -->**  **<performance ref="/sitecore/api/GraphQL/defaults/performance/standard" />**  **</jss-demo-appGraphQLEndpoint>**  **</endpoints>**  **</GraphQL>**  **</api>**  **</sitecore>**  **</configuration>** |
| --- |

Note: Above config is just a sample and not for copying for your app. Generate the config for your app and compare with this example. It is just for reference.

* Review the application config patch file(s) in ***"/sitecore/config"*** to ensure that it is configured appropriately for your Sitecore installation. If required review it with Platform team.

**Step 5: Platform team needs to deploy following configs to Sitecore server. This is a one time setup done by Platform team for each app that needs to be deployed into Sitecore.**

1. jss-demo-app.config
2. jss-demo-app.deploysecret.config

**Alternatively you can use following command to deploy the configs if you are setting up locally: “`jss deploy config”**

**Further changes to these configs would be done via devops pipelines to which tenants would be given access and instructions**

**Step 6: Deploy your app to sitecore using**[**jss cli**](https://jss.sitecore.com/docs/fundamentals/cli)**(as a one time step, later you would be deploying only components/templates etc.)**

Use following command to deploy the app to Sitecore:

| **$ jss deploy items --includeContent --includeDictionary --appName <appName>** |
| --- |

**Here:**

* *<appName>* refers to the app name you are deploying. e.g. jss-demo-app.
* *--includeContent* is an instruction to the 'jss deploy app' command to include the content data locally available.
* *--includeDictionary* is an instruction to the 'jss deploy app' command to include the available dictionary items.

**Example:**

| ***$ jss deploy items --includeContent --includeDictionary --appName jss-demo-app*** |
| --- |

***"jss deploy app"***is shorthand for ***"jss deploy items && jss deploy files"***. Used to deploy both the app's manifest items (routes, templates, etc) and a production build of the app's code. ***jss deploy app*** accepts all options supported by ***jss deploy items*** or ***jss deploy files***.

**Step 6: Start running connected mode via**[**jss cli**](https://jss.sitecore.com/docs/fundamentals/cli)

Use the following command to start the connected mode:

| **$ jss start:connected**  **OR**  **$ npm rum start:connected** |
| --- |

**Step 7:  Once you are done testing the local setup, you can start with setting up rendering host locally. Run the following command to get a rendering host build:**

| **$ npm rum build:rendering-host** |
| --- |

**You can find the commands in package.json file:**

**Text

Description automatically generated**

**This will give you a build generated specifically to run the rendering host. See example below.**

Graphical user interface, application

Description automatically generated

**Step 8:  Create a file named “http-renderer-no-tunnel.js” under “scripts“ folder.**

**Step 9: Add following content in the “http-renderer-no-tunnel.js” file.**

| **const {**  **startRenderingHostServer,**  **getDefaultAppInvocationInfoResolver,**  **} = require('@sitecore-jss/sitecore-jss-rendering-host');**  **const path = require('path');**  **const express = require('express');**  **/\* eslint-disable no-console \*/**  **try {**  **// const buildArtifactsPath = path.resolve(\_\_dirname, '../build');**  **startRenderingHostServer({**  **port: 5000,**  **host: '0.0.0.0',**  **appInvocationInfoResolver: getDefaultAppInvocationInfoResolver({**  **appPathResolver: (requestJson) => {**  **return path.resolve('./build-rendering-host/server.bundle');**  **},**  **}),**  **hooks: {**  **beforeServerStarted: (server) => {**  **server.use(**  **'/static',**  **express.static(path.resolve(\_\_dirname, '../build-rendering-host/static'), {**  **fallthrough: false, // force 404 for unknown assets under /dist**  **})**  **);**  **},**  **},**  **});**  **} catch (error) {**  **console.log(error);**  **}** |
| --- |

**Step 10: Update the “scripts” under package.json to add the command to start rendering-host**

| **"scripts": {**  **"start:rendering-host-notunnel": "cross-env-shell NODE\_ENV=development JSS\_RENDER\_ENGINE=http \"node scripts/http-renderer-no-tunnel.js\"",**    **}** |
| --- |

**Step 11: Update the site definition and config file to add the http based rendering engine and url. Edit the *jss-demo-app.config (found under "/sitecore/config")*.** **Add following in the config under <javaScriptServices>/<apps>/<app>:**

| ***serverSideRenderingEngine="http"***  ***serverSideRenderingEngineEndpointUrl="http://localhost:5000"*** |
| --- |
| ***E*xample (see highlighted):**  ***<javaScriptServices>***  ***<apps>***  ***<!--***  ***JSS App Registration***  ***The JSS app needs to be registered in order to support layout service and import services.***  ***There are many available attributes, and they inherit the defaults if not explicitly specified here.***  ***Defaults are defined in `/App\_Config/Sitecore/JavaScriptServices/Sitecore.JavaScriptServices.Apps.config`***  ***NOTE: graphQLEndpoint enables \_Integrated GraphQL\_. If not using integrated GraphQL, it can be removed.***  ***-->***  ***<app name="jss-demo-app"***  ***sitecorePath="/sitecore/content/jss-demo-app"***  ***useLanguageSpecificLayout="true"***  ***graphQLEndpoint="/api/jss-demo-app"***  ***inherits="defaults"***  ***serverSideRenderingEngine="http"***  ***serverSideRenderingEngineEndpointUrl="http://localhost:5000"***  ***/>***  ***</apps>*** |

**Step 11: Redeploy the config using “jss deploy config” command. Or copy/replace manually to your local sitecore instance path. For example: C:\\inetpub\\wwwroot\\sc93.local.com**

**Step 12: Start the rendering host and test experience editor for your app. Run the following command:**

| ***npm run start:rendering-host-notunnel*** |
| --- |
| ***This will following output and your experience editor is ready to test:***  ***C:\sitecoreFrontend\jss-demo-app>npm run start:rendering-host-notunnel***  ***> jss-demo-app@18.0.0 start:rendering-host-notunnel C:\sitecoreFrontend\jss-demo-app***  ***> cross-env-shell NODE\_ENV=development JSS\_RENDER\_ENGINE=http "node scripts/http-renderer-no-tunnel.js"***  ***Starting rendering host at localhost:5000*** |