BAMBOO RUNBOOK

CONTENTS

1. INTRODUCTION:

1.1. WHAT IS BAMBOO?

Bamboo is a continuous integration (CI) server that can be used to automate the release management for a software application, creating a continuous delivery pipeline.

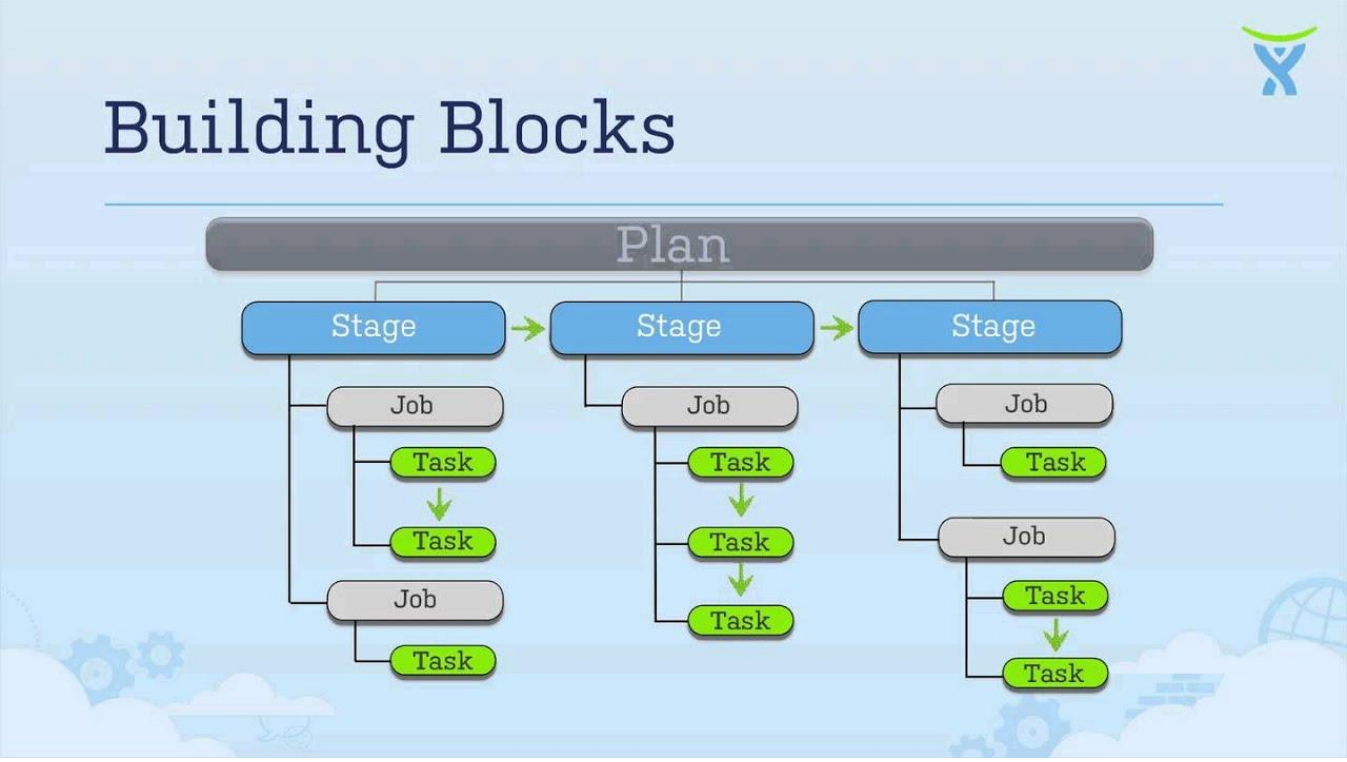
1.2. HOW BAMBOO WORKS?

Bamboo works as plan

- Which is further subdivided into stages
    - And stages has jobs
        - and further subdivided into tasks (one at a time, stop the entire build if failure occurs)

When a plan is executed

- each stage is executed in a sequence
    - Which further executes job in a simultaneously/parallel
        - Which further executes tasks in a sequence



Since jobs can be run simultaneously/parallel tasks depending on other tasks should be grouped within a job so that it executes in a sequence.

Jobs run parallel inside the stage

Stages run in a sequence:

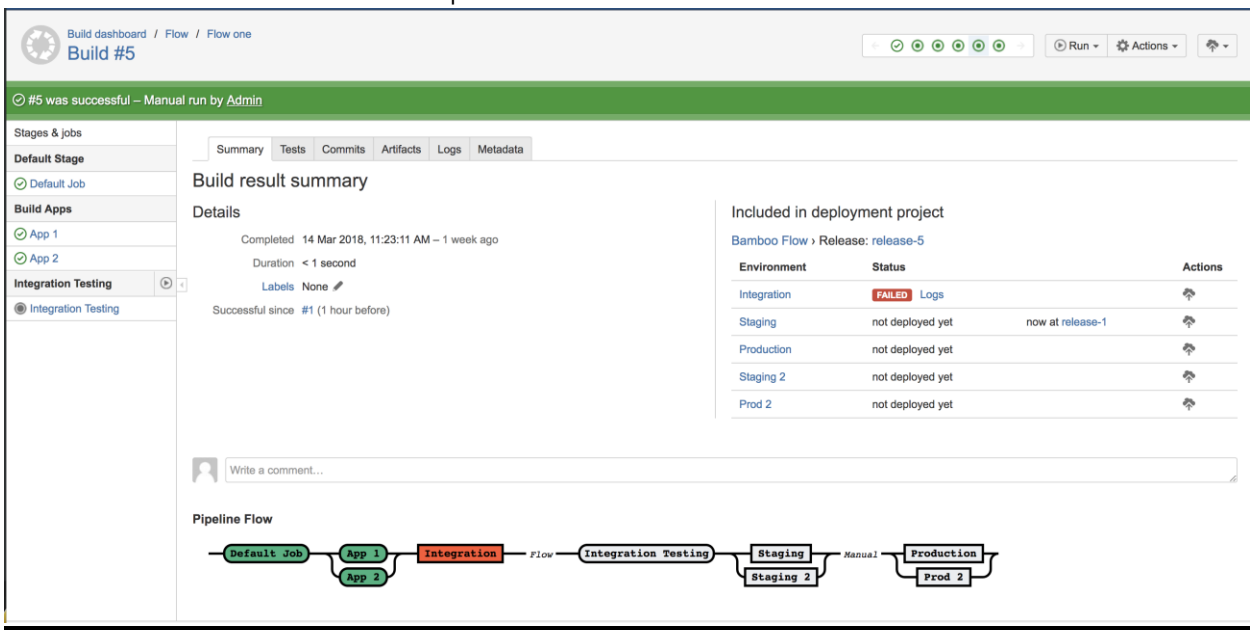
Example adding a build and deployment job in a same stage would cause jobs to perform parallel and since the deployment may execute without having the build file in it.

To avoid these jobs can be added in different stages.
A stage can be designated manual (triggered manually).

NOTE: Only moves to the next step if the previous is success.

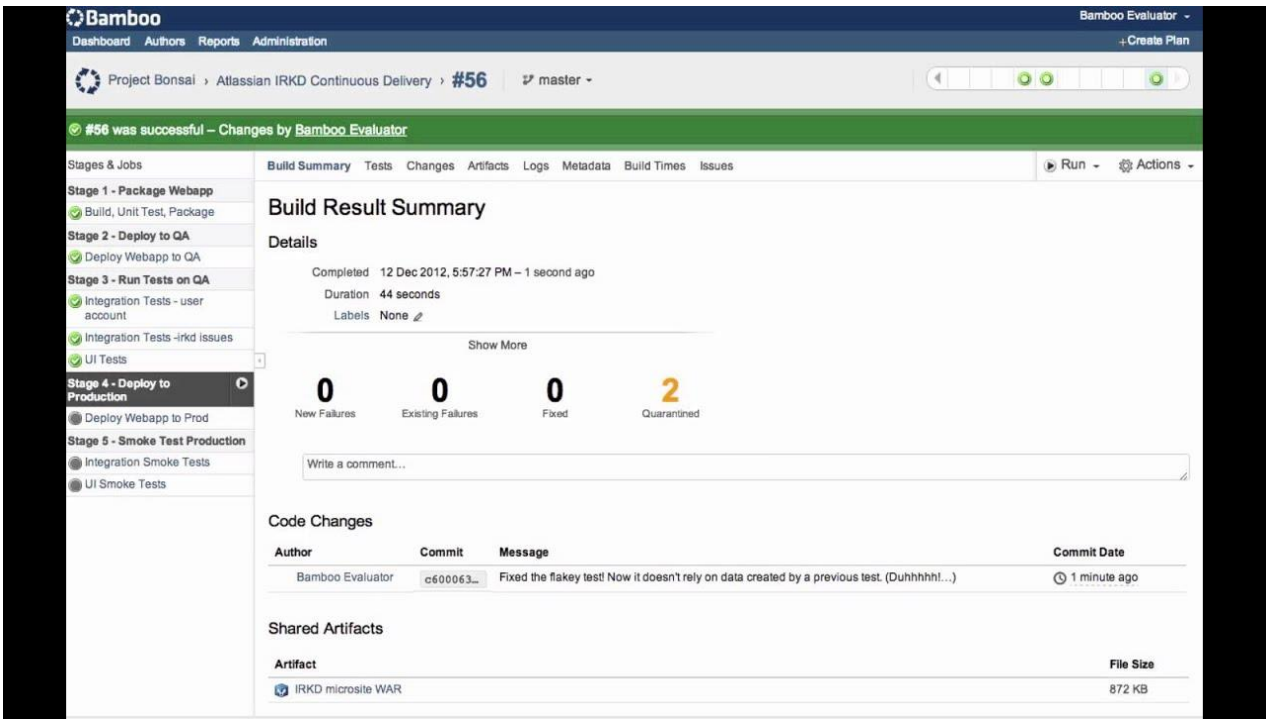
2. VISUALIZATION

Market place URL: https://marketplace.atlassian.com/apps/1218747/pipeline-flow-for-bamboo?hosting=server&tab=overview
This enables the visualization of plan in the dashboard



2.a plan visualization

3. CI/CD BAMBOO DASHBOARD SAMPLE:



3.a Project Dashboard

4. IMPLEMENTING THE PIPELINE:

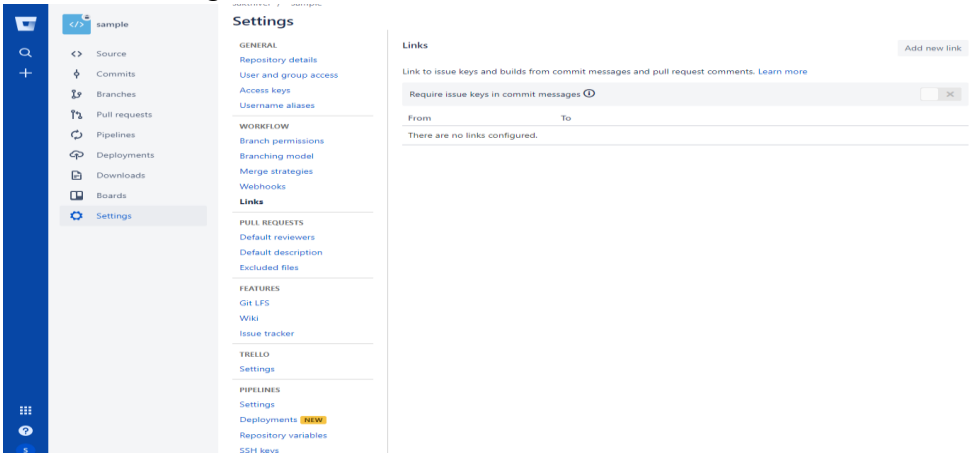
4.1. PREREQUISITES:

- A repository in Bitbucket with bamboo.yml/bamboo.yaml file/java configuration
- Jira
- Bamboo
    a. In case of yml/yaml configuration create new project with Bitbucket repo
    b. In case of java configuration not required.

4.2. INITIAL SETUPS:

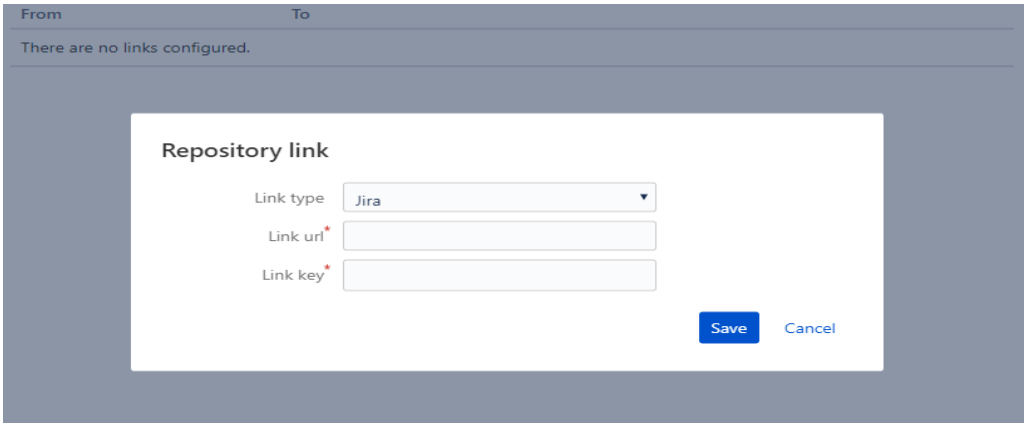
4.2.1. BITBUCKET JIRA INTEGRATION:

1. Click on Settings-> links in Bitbucket



4.2.1.a Bit bucket integration

2. Click on Add new link



4.2.1.b Link adding dialog box

3. Choose Jira and Enter the link url and link key

4.2.2. JIRA XRAY TEST RESULT DISPLAY INTEGRATION:

Integrating Jira into Bamboo using bamboo marketplace

App available in : https://marketplace.atlassian.com/apps/1216932/xray-for-jira-add-on-for-bamboo?hosting=server&tab=overview

4.2.2.1. Installation:

- Log into Bamboo instance as an admin
- Click the admin dropdown and choose Add-ons.
- Click Find new apps or Find new add-ons
  - The place where more apps and add-ons can be added to bamboo
- Locate Xray for Jira Add-on for Bamboo via search
- Click install to download and install your app.

4.2.2.2. Configuration:

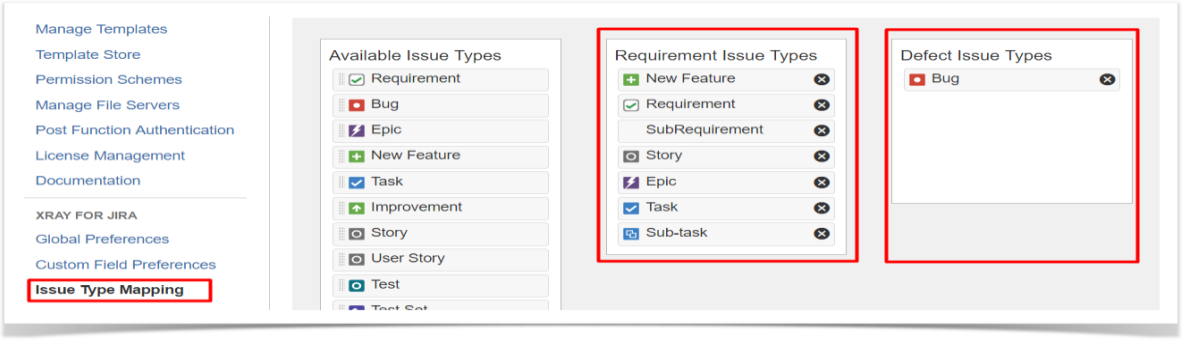
Endpoints available for xray for jira addon to import results: https://confluence.xpand-it.com/display/public/XRAY/Import+Execution+Results+-+REST

Quick Setup:

Reference: https://confluence.xpand-it.com/display/public/XRAY/Quick+Setup

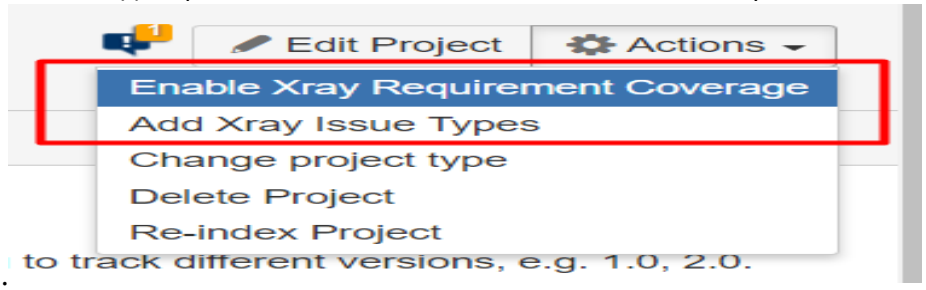
Steps:

1. In jira administration choose Add-ons tab choose Issue Type mapping and choose requirement issue type and defect issue type.



4.2.2.2.a Issue type mapping

2. Add to the project by choosing **Enable Requirement coverage** so that the requirements are covered.
3. And add issue type (i.e Test Test Set, Test Execution, Test Plan) to the project.



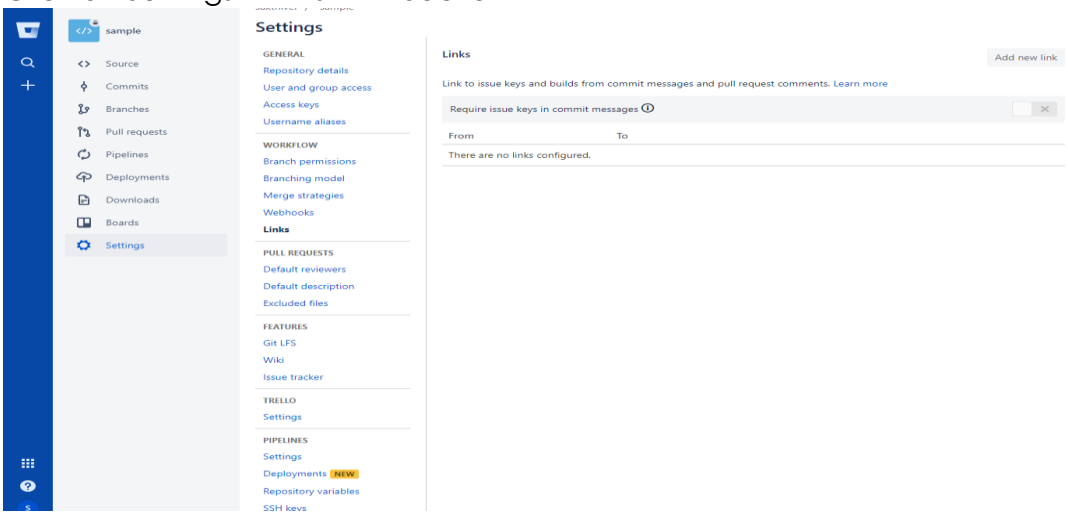
4.2.2.2.b Actions tab

4. To add on the creation of the project in the business tab choose Xray Test Project

4.2.2.2.c xray test project creation

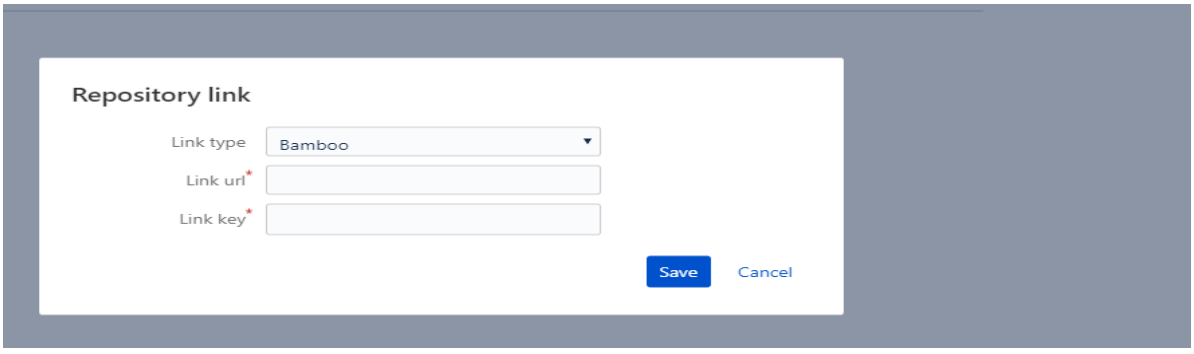
4.2.3. BITBUCKET BAMBOO INTEGRATION:

1. Click on Settings-> links in Bitbucket



4.2.3.a bitbucket settings

2. Click on Add new link



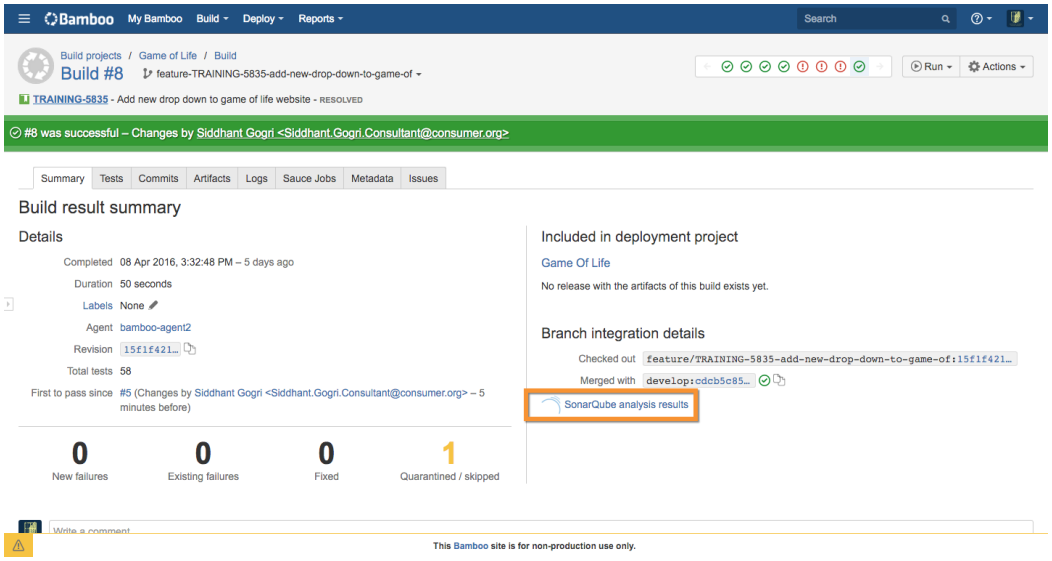
4.2.3.b Repository link dialog box

3. Choose type as Bamboo and enter the link url and the link key

4.2.4. SONARCUBE INTEGRATION:

Reference: https://www.addteq.com/blog/2016/04/automated-code-quality-analysis-integrating-sonarqube-with-Bitbucket-and-bamboo

In Bamboo you to add the sonarQube maven task within the build job in order for sonarQube to trigger an analysis. Link to sonarQube analysis is visible in the Bamboo results summary page.



4.2.4.a dashboard with sonarqube results link

4.3. BUILDING A PLAN (CONFIGURATION AS CODE):

Bamboo always looks into YAML spec first and if yamlspec is not found then it looks into javaspec

Official and Detailed documentation available at :
https://docs.atlassian.com/bamboo-specs-docs/6.9.0/specs.html?yaml#yaml

4.3.1. CONFIGURATION AS CODE USING YAML:

Bamboo looks for configuration files in

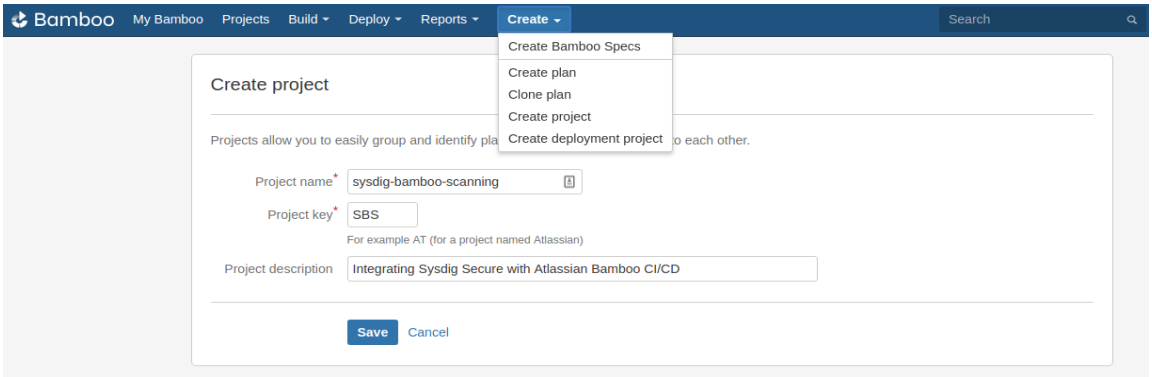
bamboo-specs

|

|---bamboo.yml/bamboo.yaml

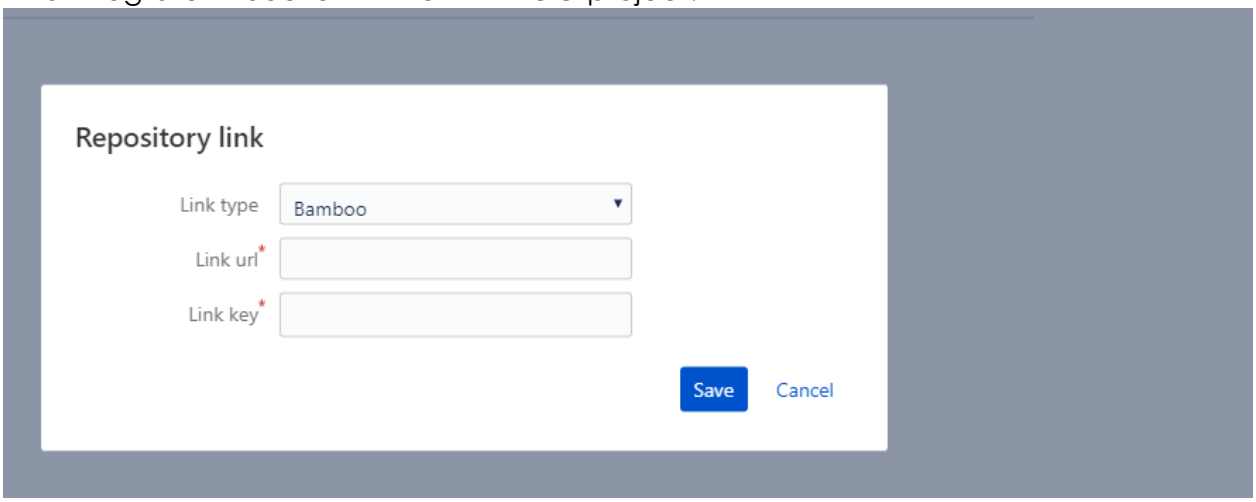
METHODOLOGY:

NOTE: It is not possible to create new project with YAML spec. so, a project should be created in bamboo before writing a yaml file.



4.3.1.a Project Creation Page

And integrate Bitbucket in the BAMBOO project.



4.3.1.b add repository dialog box

Starting from Bamboo 6.9 a version of YAML was introduced.
Starts with

```
---
version: 2
```

4.3.1.1 Creating a plan:

NOTE: Each plan requires different YAML files

- On multiple yaml files:
  - The bamboo.yml file is considered as the root file.
  - The structure is as

```
bamboo-specs
  ├── bamboo.yaml
  ├── planA.yaml
  └── planB.yaml
```

- Contains a default stage but can be used to group jobs into multiple stages.

```
plan:
  project-key: ROCKET #project key
  name: Launch Rocket #project name
  key: LAUNCH #link key
```

```
stages:
  - First Stage:
  #stages for the plan
```

4.3.1.2. Creating a stage:

- Each stage will contain a set of jobs that is executed in parallel

```
stages:
  - First Stage
          jobs:
            - First Stage Job Task
             # contains jobs
  - Second Stage:
          final: true # will be executed even if previous result fails
          jobs:
            - Second Stage Job Task
  - Third Stage:
          manual: true # waits for manual trigger
          jobs:
            - Third Stage Job Task
```

4.3.1.3. Creating a job:

- Processes a series of tasks in a sequence.

```
First Job:
 tasks:
 - Task to be executed
```

4.3.1.4. Creating a Task:

- Small unit of work

```
tasks:
  - script: echo 'Hello world'
```

- Two types namely:
  - Build Tasks
    - If build tasks fails all subsequent tasks will not be executed
  - Final Tasks
    - It runs after build tasks even if the build tasks or other previous final-tasks fails or even if the build task is stopped manually.

```
tasks:
  - clean
  - script:
     - touch report.xml
final-tasks:
  - test-parser: testing
```

4.3.2. CONFIGURATION AS CODE USING JAVA:

4.3.2.1. Why JAVA

- Reuse is limited
- Validation

- Validation tools not supported
- No integration with tools
  - Like autocomplete

JAVA

Maven annotation : @BambooSpec
Bamboo looks for configuration files in
bamboo-specs
|
|---pom.xml

It is possible to create a new project with java spec:

```
Project project = new Project()
   .key("PROA") // project key
   .name("Project A")//project name
   .description("My Project A with all A-plans");//description of the project
```

To create a **plan** a **stage** is required,
To create a **stage** a **job** is required,
To create a **job** a **task** is required

4.3.2.2. Create a job:

```
// a job with build tasks and final tasks
Job job = new Job("Job", "JOB")//each jobs name
   .tasks(new VcsCheckoutTask()
      .addCheckoutOfDefaultRepository(), new ScriptTask()
      .fileFromPath("build.sh"))//build tasks
   .finalTasks(new ScriptTask()
      .fileFromPath("cleanup.sh"));//final tasks
```

4.3.2.3. Create a stage:

```
Stage stage = new Stage("My Stage")//stage name
   .description("This is a manual stage")
   .jobs(job)//adding job with job object
   .manual(true)//manual trigger
```

4.3.2.4. Create a plan

Now a plan can be created with the **stage** and the **project:**

```
Plan plan = new Plan(project, "My Plan One", "PLAN1")//add plan to project
      .description("This is an example of a plan")
      .enabled(true)//initially have the plan enabled
      .stages(stage);//adding stage to the plan
```

5. PIPE LINE NOTIFICATIONS:

Reference : https://confluence.atlassian.com/bamboo/notifications-289276969.html
Notifications can be sent to recipients when an event occurs, Recipients can also be grouped.

5.1. YAML:

```yaml
Environment:
  notifications:
    - recipients:
        - users:
          - admin # roles of users
          - joe
        - emails:
          - admin@gmail.com # email ID's of user
          - joe@gmai.com
      events:
        - deployment-failed # Events for notification
        - deployment-finished
        - deployment-started-and-finished
    - events: deployment-failed
      recipients:
        - groups:
          - bamboo-admin# Groups for notification
          - bamboo-users
```

5.2. JAVA:

Custom Notification:

```
Notification notification = new Notification()
   .recipients(new AnyNotificationRecipient(new AtlassianModule("my-notification-recipient:plugi
n"))
      .recipientString("recipient-configuration"))
   .type(new AnyNotificationType(new AtlassianModule("my-notification-type:plugin"))
      .conditionString("type-configuration"));
```

Plan Notification:

```
Plan plan = new Plan(project, planName, planKey)
   .notifications(new Notification()
      .recipients(new UserRecipient("bob"))
      .type(new PlanStatusChangedNotification()), new Notification()
      .recipients(new ResponsibleRecipient())
      .type(new JobFailedNotification()));
```

Deployment Notification:

```
Environment environment = new Environment("QA")
   .notifications(new Notification()
      .recipients(new GroupRecipient("admins"))
      .type(new DeploymentFailedNotification()), new Notification()
      .recipients(new EmailRecipient("dev@group.com"))
      .type(new DeploymentFinishedNotification()));
```

6. ENVIRONMENT:

An environment represents the servers or groups of servers where the software release has been deployed to, and the tasks that are needed for the deployment to work smoothly. You can also define when and to whom notifications should be send

6.1. YAML:

```
environments:
 - Test # groups
 - QA
 - Prod

Test: # tasks for groups
 - tasks:
  - clean
  - artifact-download:
   - destination: workdir
```

6.2. JAVA:

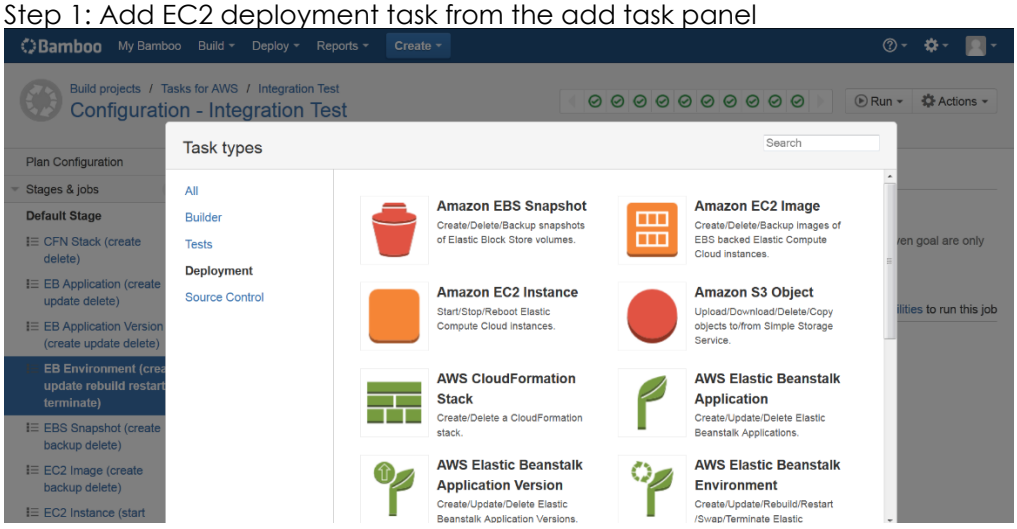
```
Environment environment = new Environment("QA")
  .tasks(new ArtifactDownloaderTask()
    .artifacts(new DownloadItem()
      .allArtifacts(true)), new ScriptTask()
    .inlineBody("echo hello"))
  .triggers(new ScheduledDeploymentTrigger()
    .scheduleOnceDaily(LocalTime.MIDNIGHT))
  .requirements(new Requirement("isLocalAgent"))
  .notifications(new Notification()
    .type(new DeploymentFinishedNotification())
    .recipients(new UserRecipient("aUser")));
```

7. Deployment to aws:

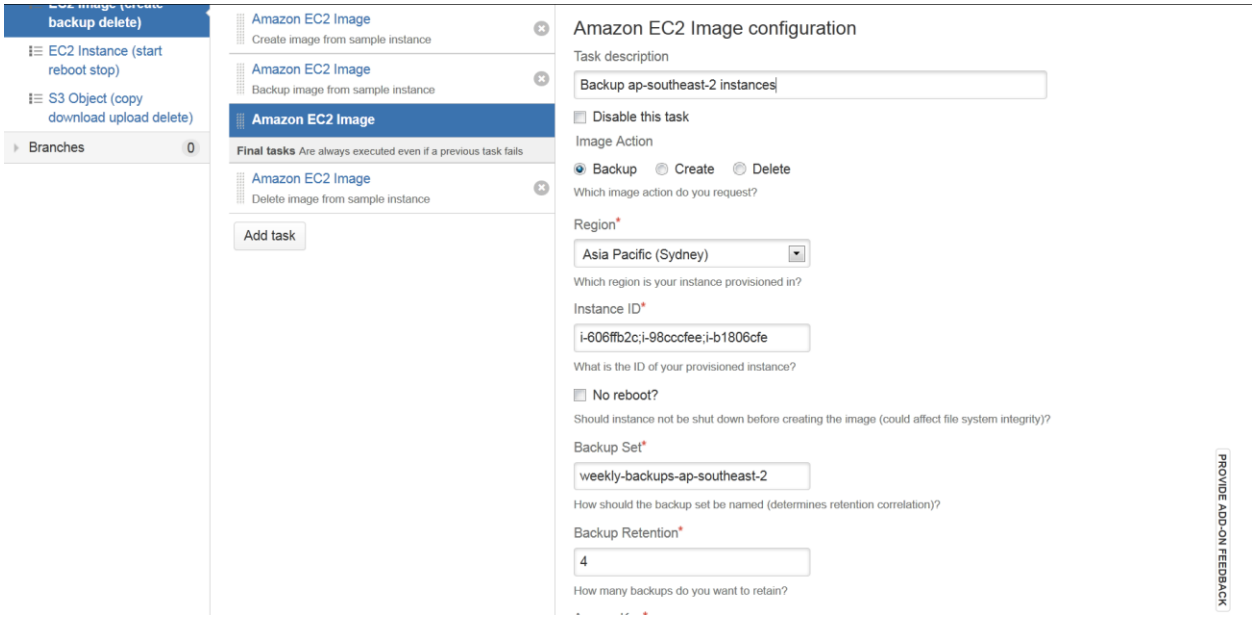
Marketplace URL=https://marketplace.atlassian.com/apps/1211585/tasks-for-aws-bamboo?hosting=server&tab=overview

Add plugin from the marketplace(using marketplace url)

Step 1: Add EC2 deployment task from the add task panel

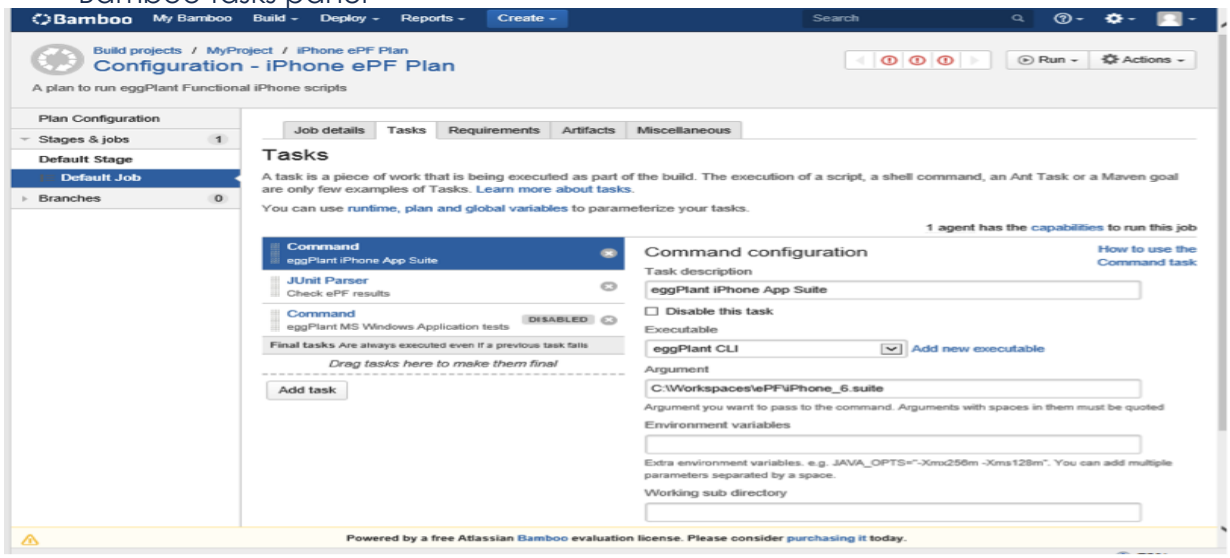


Step 2: Select the required action type and fill the credentials of the aws



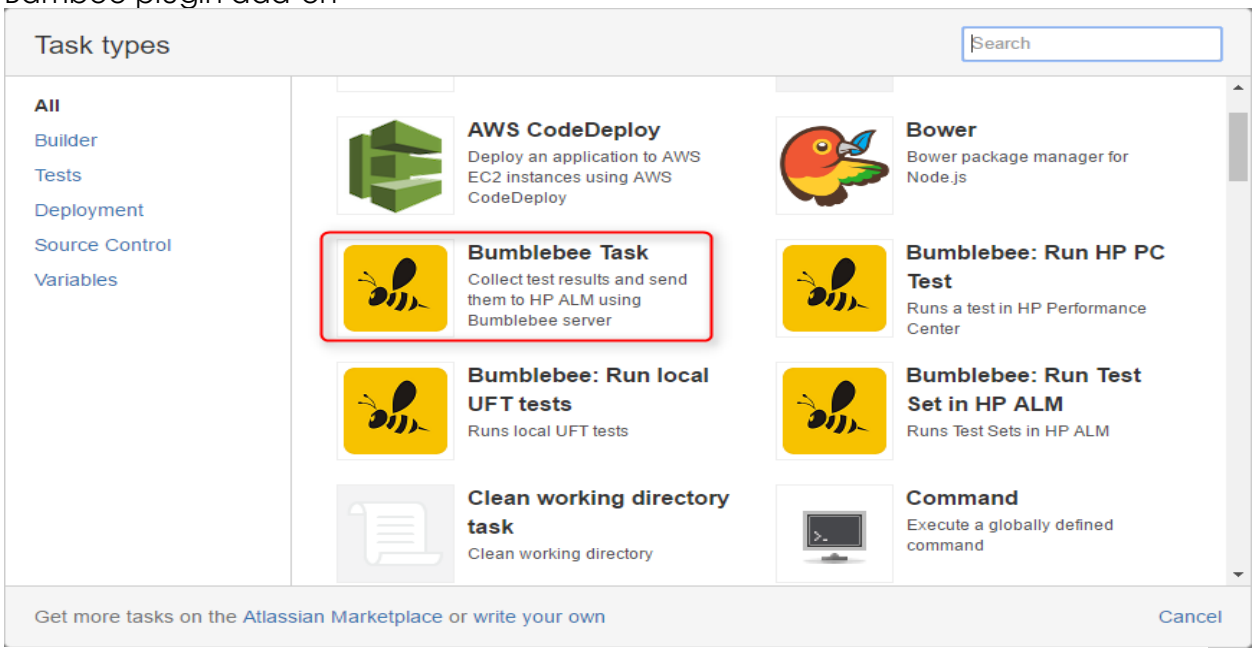
8. SAMPLE PLAN IMAGES:

Bamboo tasks panel



7.a bamboo tasks panel

Bamboo plugin add-on



7.b bamboo task dialog box

9. References :

Sample bamboo java spec = https://confluence.atlassian.com/bamboo/tutorial-create-a-simple-plan-with-bamboo-java-specs-894743911.html
YAML reference = https://confluence.atlassian.com/bamboo/bamboo-yaml-specs-938844479.html
Bamboo spec reference = https://docs.atlassian.com/bamboo-specs-docs/6.9.0/specs.html?yaml#yaml
Adding LDAP to Bamboo= https://confluence.atlassian.com/bamboo/integrating-bamboo-with-ldap-289277210.html
Bamboo deployment plan using nexus= https://community.atlassian.com/t5/Questions/Bamboo-Deployment-Plan-Using-Nexus-or-other-maven-repo-for/qaq-p/418020
Version control systems for Bamboo= https://confluence.atlassian.com/bamboo/linking-to-source-code-repositories-671089223.html
Cloud Deployment (Heroku)= https://www.youtube.com/watch?v=rG-XxVYNS4c
Provisioning aws = https://utoolity.atlassian.net/wiki/spaces/TAWS/pages/55836677/Provisioning+with+AWS+CloudFormation
Aws Deployment= https://www.youtube.com/watch?v=P7i01eqmzrs