Chapter 4 - Base Project Setup

This chapter explores how to set up your project management and version control integration, ensuring proper tracking from requirement to deployment.

Initial Project Creation

SFDX Project Setup

Create Base Project

```
sf project generate
    --name "your-project-name"
    --template standard
    --namespace "your_namespace"  # if applicable
    --default-package-dir force-app
```

Required Project Structure

```
your-project-name/
├── config/
│   └── project-scratch-def.json
├── force-app/
│   └── main/
│       └── default/
├── scripts/
│   ├── apex/
│   └── soql/
├── .forceignore
├── .gitignore
```

```
├── package.json
└── sfdx-project.json
```

Configuration Files Setup

`.forceignore` Essential Entries

```
# Standard Salesforce ignore patterns
**/.eslintrc.json
**/.prettierrc
**/.prettierignore
**/.sfdx
**/.sf
**/.vscode
**/jsconfig.json

# Package directories
**/force-app/main/default/profiles
**/force-app/main/default/settings
```

`.gitignore` Essential Entries

```
# Salesforce cache
.sf/
.sfdx/
.localdevserver/

# VS Code IDE
.vscode/

# System files
.DS_Store
*.log
```

Bitbucket Repository Integration

Initial Repository Setup

In Bitbucket:

```
- Create new repository
- Repository name: your-project-name
- Access level: Private
- Include README: Yes
- Include .gitignore: No (we'll use our own)
```

Linking Local Project to Remote Repository

Initialize Git Repository

```
cd your-project-name
git init
git add .
git commit -m "Initial project setup"
```

Link to Bitbucket

```
git remote add origin https://bitbucket.org/your-workspace/your-project-name.git
git push -u origin main
```

Branch Protection Rules

Configure in Bitbucket Repository Settings:

YAML

```
Branch Permissions:
  main:
```

```
    - Require pull request approvals
    - Minimum approvers: 2
    - Block force pushes
  develop:
    - Require pull request approvals
    - Minimum approvers: 1
    - Block force pushes
```

Project Configuration Files

`sfdx-project.json` Configuration

JSON

```
{
    "packageDirectories": [
        {
            "path": "force-app",
            "default": true,
            "package": "your-project-name",
            "versionName": "Version 1.0",
            "versionNumber": "1.0.0.NEXT"
        }
    ],
    "namespace": "",
    "sourceApiVersion": "60.0"
}
```

`project-scratch-def.json` Base Configuration

JSON

```
{
    "orgName": "Your Project Name",
    "edition": "Enterprise",
    "features": ["EnableSetPasswordInApi"],
    "settings": {
        "lightningExperienceSettings": {
            "enableS1DesktopEnabled": true
        },
        "securitySettings": {
```

```
            "passwordPolicies": {
                "enableSetPasswordInApi": true
            }
        }
    }
}
```

Post-Setup Verification

Run these commands to verify setup:

Bash

```
# Verify SFDX project
sf project verify

# Verify Git setup
git remote -v

# Verify Bitbucket connection
git fetch origin

# Verify branch protection
git push origin main --dry-run
```

JIRA Configuration

- Create a new project
- Configure the following required elements:
  - Epic issue type
  - Story issue type
  - Bug issue type
  - Task issue type
  - Custom fields for Salesforce metadata tracking

Required JIRA Workflow States

Text Only

```
Backlog -> In Progress -> In Review -> Ready for Deploy -> Done
```

Bitbucket Integration

- Link your JIRA project to Bitbucket repository
- Configure repository access rights
- Setup branch policies:
  - `main` - protected, requires PR
- `develop` - protected, requires PR
- `feature/*` - development branches
- `hotfix/*` - emergency fixes

Work Segmentation

Story Creation Rules

Stories should be:
- Independent (can be deployed alone)
- Small enough to be completed in 1-3 days
- Tagged with proper metadata types
- Linked to an Epic

Required Story Fields

- Epic Link
- Acceptance Criteria
- Metadata Types
- Development Notes
- Test Cases

Integration Setup

JIRA to Bitbucket Connection

1. In JIRA:
   - Navigate to Project Settings
   - Enable "Development" integration
   - Link to Bitbucket repository

2. In Bitbucket:
    - Configure branch policies
    - Setup automatic JIRA issue transitions
    - Enable smart commits

Commit Message Format

Text Only

```
[PROJ-123] Brief description


- Detailed changes
- Impact on existing functionality
- Related configuration
```

Pipeline Configuration

Get the bitbucket-pipelines.yml file

Integrate it and set up the variables

Automation Rules

JIRA Automation

- Create branch from ticket
- Update ticket status on commit
- Link PR to ticket
- Transition on successful deployment

Bitbucket Pipelines

- Trigger on PR creation
- Run validation suite
- Deploy to appropriate environment
- Update JIRA ticket status

Security and Access

Required Team Roles

- Project Admin (JIRA + Bitbucket)
- Development Team (restricted repository access)
- Release Manager (deployment rights)
- QA Team (environment access)

Access Matrix

Text Only

```
Role          | JIRA |  Bitbucket |  Salesforce
Project Admin |  Admin|  Admin     |  System Admin
Developer     |  Write|  Write     |  Developer
QA           |  Write|  Read      |  Read-only
```

Remember that this setup needs to be done only once per project, but maintaining the discipline of following these structures is crucial for successful CI/CD implementation.

The key to success is ensuring that:
1. Every piece of work has a ticket
2. Every commit links to a ticket
3. Every deployment is traceable
4. All changes are reviewable

This structured approach ensures that your project management directly ties into your deployment pipeline, making it easier to track changes and maintain quality throughout the development lifecycle.

Revision #2
Created 4 July 2024 15:44:44 by Windyo
Updated 7 November 2024 14:42:12 by Windyo