Technology Stack & Engineering Best Practices

Starter Architecture

1. Development Environment

Approach: Containerization

Technologies - Docker Compose

Syntax - Yaml

Language - Go

Starter Architecture

2. Back-end API

Approach: S.O.L.I.D Principles

Technologies - Laravel

Syntax - Object-Oriented design (OOD)

Language - PHP

Starter Architecture

3. Front-end SPA/SSR

Approach: Composition API

Technologies - VueJs/Nuxt

Syntax - Typescript

Language - Javascript

Starter Architecture

4. Server Provisioning

Approach: Configuration Management

Technologies - Ansible

Syntax - YAML

Language - Python

Starter Architecture

5. Release Engineering

Approach: Continuous Integration and Continuous Deployment (CI/CD)

Technologies - Jenkins, Docker

Syntax - Jenkinsfile (declarative or scripted pipeline syntax)

Language - Groovy (used in Jenkinsfile)

Starter Architecture

6. Infrastructure as Code

Approach: Cold Start

Technologies - Terraform

Syntax - HashiCorp Configuration Language (HCL)

Language - N/A (HCL is a domain-specific language for Terraform)

Engineering Best Practices

Phase 1

PLAN

Processes :

1. Discovery/Product Roadmap (User Story Creation)
2. Release Planning (Effort Breakdown according to SCRUM, Sprint Planning)
3. Agile Life Cycle (BRD Document Drafts, Client Meetings)
4. Management
5. QA Strategy
6. Compliance requirements / Industry Standards

Tools :

- Redmine
- Slack

Engineering Best Practices

Phase 2

BUILD AND DEVELOPMENT

Processes :

1. Local & Cloud Environments + Configuration Management (Git Repository extending the Starter Kit integration)
2. Branching Strategy (Git Flow model)
3. Coding Standard / Practices
   * Infrastructure as Code for DevOps using Ansible and Docker
   * S.O.L.I.D Principles for Back-end using Laravel
   * Composition API for Frontend Javascript using Typescript
   * HTML5 Semantic Elements following BEM (Block, Element, Modifier) Methodology for HTML Layouting
   * Mobile-First & Responsive Design following CSS Grid and Flexbox systems for CSS Layouting using SASS)
4. Code Control / Analysis

Tools :

- Github
- Docker
- Docker Compose
- Ansible
- Laravel
- Vue
- Nuxt
- Sass

Engineering Best Practices

Phase 3

CONTINUOUS INTEGRATION

Processes :

1. CI/CD Pipeline (Ansible playbook used for provisioning a Jenkins server responsible for running CI/CD Pipelines for all needed environments)
2. Build Environment (CI/CD tech stack and implementation)

Tools :

- Docker
- Docker Compose
- Jenkins
- Ansible

Engineering Best Practices

Phase 4

TEST

Processes :

1. QA Workflow
2. Functional Testing
3. Sanity/Regression Testing
4. Performance Testing
5. Penetration Testing & Security Audits

Tools :

To Be Decided

Engineering Best Practices

Phase 5

DEPLOY AND RELEASE

Processes :

1. Deployment Strategy (Ansible playbook used for deploying builts to their respective hosting resources)
2. Release Management & Dashboards
3. Automate Deployments / Rollbacks
4. GitOps Workflows for AWS Reference Architecture (Comprehensive Cold Start Infrastructure as Code) **To Be Decided**

Tools :

- Kubernetes
- Helm
- Helmfile
- Atmos
- Ansible

Engineering Best Practices

Phase 6

SITE RELIABILITY ENGINEERING

Processes :

1. Code Quality Monitoring
2. Dashboarding
3. Notifications
4. Metrics, APM, Logs, Tracing & Performance Monitoring

Tools :

To Be Decided