

Aglie assignment-JiaJiTao

by Jiaji Tao

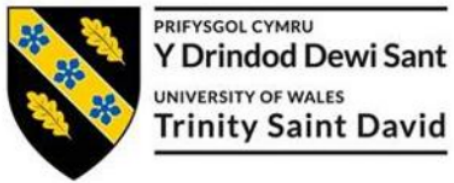
Submission date: 15-Dec-2022 04:44PM (UTC+0000)

Submission ID: 192581105

File name: 40680_Jiaji_Tao_Aglie_assignment-JiaJiTao_460463_1030388259.docx (645.78K)

Word count: 6572

Character count: 35298



Agile Development and Software Dev Ops

Name: JiaJi Tao

Student number: 2200521

1. TASK1

1.1 Agile development theory

Scrum originated from the term football and was introduced by Sutherland and Schwaber in 1995 in the United States. It is an iterative, incremental development framework. Agile development is a guiding philosophy without a clear and specific development process, while Scrum is a specific way to guide software development, including the division of roles for team members, development process and requirements analysis. Scrum includes a series of practices and a process framework with predefined roles, an iterative and incremental development approach, a high level of collaboration between internal teams and users, a focus on delivering high-value software products to customers, and a continuous improvement of team capabilities through the development process flow and post-sprint reflection and summarization. team capabilities, it is well positioned to respond to rapidly changing requirements and deliver new versions frequently. Based on the Scrum agile development framework, the project development cycle is divided into several small iteration cycles. Each small iteration cycle is called a sprint, and a cycle usually takes 30 days. In Scrum, product requirements management is performed using a product Backlog, which defines all the tasks of the product, including functional and non-functional tasks. After response improvement based on Scrum methodology, it is applied in software project management to improve the overall enthusiasm and efficiency of the team's work. the Scrum development model is shown in Figure 1.

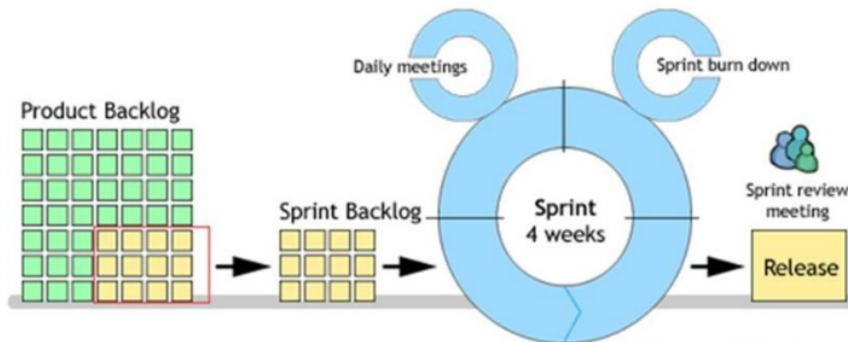


Figure 1 scrum development model

Agile project management theory has been developed for more than a decade and is being introduced by more and more software companies. In software project management agile development practice, Scrum development method is the most widely used and has become the first choice of most agile development teams. Based on the development and testing work in the R&D centre of a communications company, Chen and Luo proposed an improvement plan for the Scrum methodology, including the improvement of the daily meeting model, the improvement of the pair programming approach, the avoidance of formal review meetings, and the insistence

on continuous integration and test-driven methods. Through the implementation of the Scrum improvement plan, the productivity of the team and the motivation and efficiency of the team members have been effectively improved.

Xu Wei and Wang Haoning, based on their research on the combination of CMMI standards and agile development, point out that by combining the two, CMMI standards can better cope with large software development, and Scrum development can be applied in a larger range of medium and large software projects to better meet user needs. Many people believe that Scrum development is better suited to small software development, while CMMI is better suited to medium to large software development, and that the two are in opposition to each other, and that a mixture of them will cancel out each other's effectiveness. However, this confrontational philosophy is not only wrong, but also prevents the development of high quality software products in the short term. To get good results, it is best to create a hybrid CMMI and Scrum model.

Wang Xiaolin and Zeng Hongwei (2019) conducted a study on the optimisation techniques for regression testing in agile development environments, proposing new regression testing optimisation techniques in Scrum development environments based on the problems of large test set base and long time spent in the testing process. By splitting regression tests, combining agile sequencing and agile selection methods, a regression test optimisation model is designed. Finally, the effectiveness of the optimisation scheme is verified by selecting case projects for testing and recording the implementation results to prove the optimality of the scheme.

The foundation of Scrum theory provides a faster and more efficient development method for the software development industry. Before the Scrum development framework was proposed, software enterprise projects were developed using the waterfall model. The linear development model of each link makes the whole project development process slow and unpredictable. In contrast, Scrum has the advantages of lightweight and high flexibility. After the introduction of the agile framework, some companies that pursue innovation and have strong corporate strength began to try and apply it. Scrum always adheres to the three core principles of transparency, detection and adaptation in the implementation process. Transparency refers to that in the development process, the development progress and output of the whole project are open and transparent, which can be seen by all project related personnel. High transparency also provides a basis for real-time detection. The consistency between the project implementation progress and the project schedule can be detected. Once it is found that the project implementation deviates from the project objectives, it can be adjusted immediately to minimize the change cost of the project. The high transparency of the Scrum development model is mainly due to its meeting mechanism, such as the daily standing meeting and sprint planning meeting.

1.2 Key concepts involved in Agile

Before evaluating the implementation of these concepts in Scrum, it is important not to overlook the flow of user stories throughout the process, and first we need to understand the concept of user stories.

A user story is a short description of a customer's needs recorded in an agile tool. It has no specific criteria and can be used to record requirements depending on whether the mission is an actual need, the key is to describe the business and customer needs concisely and precisely. User stories are usually documented in the following format: User Story = Who + What + Value, who is going to use it, what activity is going to be done, why it is being done and what value it will bring. A quality user story is recommended to contain elements such as a name, description, time estimate, and priority. User stories can be provided by users, customers, the product director and members of the working team. The product director is responsible for story capture, production and maintenance, and for adjusting user story priorities based on iterations and realities.

A case study to understand these concepts, Company D was incorporated in June 2016, the existing company employs about 150 people, the company's headquarters is in Guangzhou, the main body of technology-related departments in Hangzhou, the main business is to provide cross-border payment collection service platform for major e-commerce platforms or financial institutions. Under the existing organisational structure of Company D, several business units are divided according to the different responsibilities of the departments. The CTO line (Product Development Department) includes the Delivery Business Department, the Technology Development Department and the Business Services Department, whose main responsibility is to be responsible for the development and operational management of Company D's software products. The specific implementation of software development projects is completed by the Software Development Department, which is divided into development groups, UI groups, testing groups, project management support groups and product groups according to the features required for the projects. In the organisational structure of Company D, the CTO, as the overall head of the company's R&D operations, coordinates the development, management and operation of all the company's projects and reports directly to the company's founder. The technical R&D department is the core department of the company and the entire technical line, and is divided into development, UI, testing, project management support and product groups, each with a supervisor. The problems of project management in Company D are mainly in four areas: requirements acquisition, R&D, testing and organisational structure. The 10 problems are as follows: frequent requirement changes, unclear requirements, less communication with users, inflexible development process, low customer involvement in the development phase, non-transparent development progress, poor internal communication, late involvement in testing, and no user involvement in the testing process.

1.2.1 Fast short iterations and incremental delivery

The Scrum development process is iterative, with projects split into iterative cycles depending on size, each iteration should not be too long, usually between 1-4 weeks. Each iteration cycle involves a complete process of requirements communication, requirements analysis, requirements design, code writing, testing and delivery, with a sprint review meeting at the end of an iteration to demonstrate the functionality to customers and collect customer feedback. Unlike the waterfall development model, where a list of features is developed and delivered to the customer in one go, Scrum development incorporates a sprint development plan that prioritises the features identified with the customer, prioritises the completion and delivery of a list of features with high market value and clear requirements, and prioritises the delivery of some of the available features to run after each sprint cycle, followed by Incremental deployments will be carried out on the basis of the previously completed versions, resulting in the delivery of a fully functional software product that meets customer expectations.

Agile teams do exactly the same development work as Waterfall teams, but they do things very differently. The agile development cycle uses the same functions as the waterfall approach: requirements gathering, design, coding and testing. In simple terms, the difference between agile and waterfall development is that waterfall development has to complete the current step before moving on to the next step. This is not the way of an agile team, which does a little requirements gathering, a little design, coding and testing, and finally delivers a little value to the customer. The team then repeats the process week after week, improving and adjusting the process as the work progresses until the project is complete. Incremental iterative development changes not just the point at which things are done, but also the way they are done. Agile iterations (called 'sprints' in scrum) are not miniature waterfalls, and there are really no steps in the agile process. Agile development is a holistic process where testing, design, coding and requirements gathering are fully integrated processes that depend on each other. For example, testing is integrated into the design process. Requirements are not simply gathered, but instead require an ongoing process of communication between the team, the product owner and the customer to build up a deep and detailed shared understanding of the requirements. The core idea of the agile approach is to deliver business value quickly, in the form of working software, but also to deliver value continuously in regular increments. As we will see in subsequent chapters, the benefits to the business of iterative development are not only immediately visible but also cumulative.

Combining the characteristics of Company D's project and agile development theory, the whole project was divided into several development Sprints and in each Sprint confirmation work such as user validation, design and development, and sprint review was carried out. At the end of each sprint, a second sprint is launched with a list of features prioritised by the user and coded after receiving confirmation from the client. After each iteration, a version is released if the review passes, and any remaining issues that do not pass the review are moved to the next sprint along with

the next sprint list, and so on.

1.2.2 Emphasis on face-to-face communication and collaboration

Scrum places great emphasis on the development team. Agile teams emphasise the importance of people, with no hierarchy of leaders and led. Scrum leaders are positioned as servant leaders, providing an agile development environment for developers. A self-managed agile team has a good, harmonious and trusting working atmosphere, which increases morale and productivity and brings out the best in the team.

The Scrum agile approach places a lot of emphasis on interaction and collaboration between team members. Traditional project teams are formed on an ad hoc basis, with people coming from different departments, and there are inevitably barriers to communication. In Company D As a start-up company, the organisational model of Company D was confusing. Scrum is a different way of organising a project team, with more emphasis on self-organisation rather than reactive deployment. There are three core roles in a Scrum team, and the responsibilities of each role should be clearly defined when forming the team.

(i) Product Leader

The product owner is responsible for the marketing and sales of the product and needs to have an in-depth knowledge of the company's products and the product industry. The product owner usually represents the interests of the client. In an agile project team, there is usually only one product owner, who is responsible for confirming and maintaining the to-do list and updating the development priorities of each module in the product feature list. In the Scrum development model, the product owner is the person closest to the customer, the link between the company and the customer, and is responsible for the product deliverables, so the product's refined to-do list is managed by the product owner, and the product's functional modules need to be accepted and confirmed by the product owner after they are developed, and delivered to the customer only after the product owner has passed the evaluation. However, the product manager does not work alone, each member of the Scrum team will try to assist the product owner and work together to deliver a more marketable product.

(ii) Scrum Master

The Scrum Master, also known as an Agile Coach, is a servant leader in an Agile team. The Scrum Coach is very familiar with the Scrum development methodology and assists the Product Owner in sorting out the product feature list during the project initiation phase, organises regular agile meetings during the development process and assists the development team in specifying sprint orders for each phase. orders. At the same time, the Scrum Master, as a service leader of the team, is responsible for the external communication of the development team, intercepting external interference with the development team and ensuring the efficiency of the development team. As Company D is still in the early stage of introducing the agile development model, the company does not have a suitable person to act as an agile coach at the moment, so it can ask the current development director to act as an agile coach temporarily, and

then arrange a suitable person to act as an agile coach after the project team has a deeper understanding of the agile development model.

(iii) Development Teams

The biggest difference between Scrum Agile teams and previous project teams is self-organisation. A self-organising

team is a combination of cross-functional all-rounders who together have all the skills required to complete a product, such as architecture building, UI design, writing programs, etc. They work on their own terms, receiving tasks based on their strengths and completing them consciously, without being directed by others. Agile team members are expected to anticipate the market needed to complete a sprint based on a to-do list checklist, and negotiate within the team itself how to accomplish it. Self-organised teams are more stable and have a clearer goal orientation than traditional ad hoc teams. Self-organisation and self-management are key features of agile teams, where members are no longer just passively assigned tasks, but actively take on tasks that combine their own technical strengths and abilities. The development team can decide how to implement incremental features on the product's feature list without the intervention of non-team members. Each member of the team makes full use of their professional skills and works together to solve the problems encountered in each Sprint. Everyone in the development team works towards the same goal, and in the process of solving the problems of each Sprint, in-depth communication takes place. Both developers and O&M staff can learn new experiences and techniques from each other, which improves their own level and breaks down the barriers of the original functional departments, increasing the cohesion of the team.

1.2.3 High level of customer involvement

Scrum development includes users as an important part of the agile team, and they play an important role in the whole process of software product development. In activities such as requirements analysis and prototyping, customers are fully involved and provide timely feedback, which largely avoids deviations in the project development process. In addition, Scrum's sprint planning meetings and sprint review meetings provide a way for customers to provide input and feedback, which greatly facilitates faster and better delivery of software products.

To establish an effective communication mechanism, Company D has prominent communication problems, mainly including communication problems with users and communication problems between members within the company. The poor communication mechanism led to a lot of untimely information transfer, the work progress could not be perceived in time and the team working atmosphere was not active. The introduction of the four meeting mechanisms commonly used in agile development, which are held throughout the development phase and each with a specific meeting cycle, not only improved the visualisation of the project progress, but also achieved timely and smooth communication between project-related personnel.

1.3 Comparison with Traditional Models

1. Team composition: In the waterfall software development process, customers and stakeholders are only present in the early requirements analysis phase of the project, and in the later acceptance phase of the project, and are not involved in the project development process. Agile development requires that customers and stakeholders are involved throughout the project g development process.

2. Development process: The waterfall software development model has a long cycle time and uses a linear development approach. Agile development uses iterative incremental development, with each iteration having a shorter cycle time.

3. Adaptation to change: The waterfall software development model has difficulty adapting to changes in user requirements. Agile development can both adapt to changes, respond quickly to changes, and give a quick solution to adjust.

4. Documentation requirements: The waterfall software development model has requirements for the amount, content and format of documentation. Agile development believes that human interaction takes precedence over documentation and processes, so documentation is simplified and human-centered.

5. Testing mechanism: In the waterfall software development model, testing intervenes late and cannot be done throughout the project cycle, while in agile development, testing and development work are synchronized and automated testing is advocated.

2. TASK2

2.1 Scrum Agile Project Schedule Estimation

2.1.1 Estimation methods

Scrum agile estimation is usually done for user stories or feature points in the requirements list. There are two common scales for Scrum agile estimation.

(1) Story points

Story points represent the amount of work required by the development team to translate user stories into software products. There is no specific requirement for the initial size of story points, but story points should be highly relative. The first is to select the simplest user story with the lowest difficulty factor and lowest risk among all the user stories to be developed and set its story point to 1. The other user stories are estimated using this as the standard to estimate the corresponding story point. The second method is to find a medium sized user story and set its story points to the middle of the evaluation range, for example, if the story point evaluation range is 1 to 10, then it can be set to 5 and the other user story points will be compared to the first user story to estimate their size.

(2) Ideal day

In software projects, the ideal day can be chosen to estimate the working hours for user stories or other tasks. When applying the ideal day to the estimation of man-hours, it is important to note that the ideal day is based on the following conditions.

- a. the user story being estimated is the only work being done during the work period
- b. the resources and other conditions required for the work are available before the work begins
- c. the work is not disturbed or interrupted by other things during the work

This is the ideal state of affairs, as there are uncontrollable external factors that make it impossible to devote 100% of the actual work time to the job, so a portion of the safety time is often added to the ideal work time estimate to cover the additional time overhead.

2.1.2 Expert estimation method

Agile software estimation is the same as the traditional software project management expert estimation method, but this method is not as effective in agile development projects compared to the former. Agile development can require multiple skills, sometimes with multiple team members working together, and these require multiple skills from the estimator, which makes it difficult to find a suitable expert who can assess all tasks.

Expert assessment is characterised by its relative speed and, where a task is performed by a single person, this method of estimation has a fairly high degree of accuracy.

The expert assessment method was used in the case study with Company D. The expert opinion on the problems of the existing project management chain in Company D included four main areas: requirements acquisition problems, R&D phase problems, testing phase problems and organisational structure problems. The following 10 problems were identified: frequent requirement changes, unclear requirements, low communication with users, inflexible development process, low customer involvement in the development phase, non-transparent development progress, poor internal communication, late involvement in testing, and no user involvement in the testing process.

2.1.3 The analogy method

Unlike traditional software schedule management estimation, agile development analogy estimation uses different user stories to compare with each other, and by comparing with a baseline user story, a more accurate estimate of the user story being compared can be derived. When making an analogy, it is not necessary to set a baseline user story, but simply to compare it with a user story that has already been estimated. Agile software progress analogies are more actionable and accurate than traditional software progress analogies.

2.1.4 Planning Poker

Plan poker, also known as agile poker, is an estimation method that combines expert evaluation methods and analogies to estimate the man-hours or relative value of software development. It is a gamified estimation technique that requires different participants to independently estimate workloads with the goal of reaching consensus. Plan Poker involves all members of the development team, including programmers, testers, UI designers, database administrators, analysts, interaction designers, etc. The product manager participates in the planning poker but does not perform the actual man-hours estimation.

The specific implementation steps are.

- (1) Estimators each receive a set of estimation cards. Each person can take a set of cards with 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100 written on them, and the numbers on the cards should be large enough to allow for estimation of all user stories
- (2) The facilitator reads out the user stories that need to be estimated
- (3) The estimator asks questions about the user stories and the product owner answers them
- (4) The estimator gives an estimate based on his or her understanding of the user story, but does not show the estimated value
- (5) When all estimators have completed their estimates, the estimator flips their card so that everyone can see their estimate
- (6) If there is a significant difference in the estimates, the high and low estimators need to explain the basis for their estimates and all members discuss them accordingly
- (7) Go back to step (1) for a new round of estimation until there is agreement or similarity

2.2 Scrum Agile Software Development Project Schedule Control

Methods and Tools

From the first article of the Agile Manifesto, "Individual interaction takes precedence over process and tools", we can see that in agile project development, human-centred, team communication and feedback take precedence over all process and tool-based methods and tools. People are the main body of the software project and are the key factor in determining the success or failure of the project. Giving full play to the role of team members, timely feedback on problems and adequate teamwork are the keys to ensuring the success of the project. Common agile progress control process methods include daily station meetings, whiteboards and burn-out charts, while WBS, milestone progress and earned value methods can also be used.

2.2.1 Daily Scrum

In the Scrum agile development process, the Scrum team holds a time-limited meeting every day, where all team members gather in a circle and talk to each other in turn about a predetermined topic, to enhance team cohesion and cooperation through mutual reporting and commitment, and to provide timely feedback on the progress of the project and problems encountered. problems.

According to the 12th annual state of agile report published by VersionOne in 2018, 90% of respondents use the daily stand-up technique in agile project management, which is the top of all agile methods. This shows that the role of the team has been raised to a high level of importance in the implementation of agile development models in major companies around the world.

In each Sprint, the daily stand-up is generally limited to 15 minutes, and to increase efficiency and save time, the meeting is strictly controlled and each team member is only required to answer three questions.

- (1) What has been accomplished since the last Scrum Daily Stand-up?
- (2) What do you plan to accomplish between this Scrum Daily and the next one?
- (3) What are the difficulties in achieving the Sprint objectives and project goals?

The meeting is held by the Scrum Master, who follows a strict flow of the meeting and limits each person to

The Scrum Master follows a strict meeting process, limiting each person's questions beyond the three, and controlling everyone's time to ensure that the overall time is kept to 15 minutes. If there are additional issues that need to be discussed, the Scrum Master will need to arrange for the relevant people to discuss them at another time after the meeting. The purpose of the daily station meeting is to ensure that the work of all team members is made public, to reflect the actual work of each individual in a timely manner, and to take the necessary action as needed to ensure that the project progresses as planned.

2.2.2 Scrum Whiteboard

When conducting an agile development planning meeting, the development

team determines the user stories that can be iterated for one Sprint, depending on their situation. Team members use the Scrum Whiteboard to have the current Sprint Backlog displayed in the whiteboard. During the daily station meeting, team members continuously update the status of the user stories on the Scrum Whiteboard according to their work progress, and the actual progress of the project is brought up in the whiteboard.

In agile software development, Scrum whiteboards are generally divided into five main categories: Story, To Do, In Progress, To Verify, Done and Hang up.

Hang up.

(1) User Story: A narrative text used to describe a user's needs (e.g. as a... I need... so that I can...)

(2) Not Started: A task that has not yet been started in the current Sprint

(3) In progress: tasks that have already started development in the current Sprint.

In the daily station meeting, team members

members need to move the corresponding unstarted tasks to the In Progress column according to what they plan to do today column

(4) To be tested: When the development task is completed, team members need to move the corresponding task to the To be tested section.

to the test section. At this point, the testers can test the user story-related functionality in this section

(5) Completed: When the task has passed the testers' tests, the testers will need to move the task to the Completed section.

If the task does not pass the test, you need to remind the developer to check and fix it

(6) Pended: Due to relevant factors, this user story task is suspended and in a The task will be reworked when the time is right

2.2.3 Burn-out charts

The burn-out diagram, also known as the burn-up diagram, is a tool commonly used in agile methods to visualise the relationship between remaining work and remaining time before the project is completed. The burn-out diagram has a Y-axis (for work) and an X-axis (for time). Ideally, as time decreases, the amount of work remaining decreases proportionally, and when the time remaining is zero, the amount of work remaining 'burns out' to zero. The burn-out diagram provides project team members and stakeholders with a real time build-up of work progress, depicting the relationship between remaining time and remaining work.

The purpose of the Sprint Burn Out Chart is to give the team a clear picture of whether they can deliver all that they have committed to in the iteration. If they can do that, great! But if they can't, the chart helps them to identify problems early on so there's still time to make adjustments. The team can also let the product owner know immediately if they find that some stories are no longer available. The product owner can then consult with the team on how to deal with the situation. The product owner may choose to abandon a story in the sprint, or may narrow down the scope of one or

more stories until the team can complete the rest. If the team realises they will finish all the stories early, they can approach the product owner for another story to continue development.

When the team is just starting out, it is normal for them to find that there are new tasks that need to be completed. When they find these tasks, they have to estimate their size and add them to the sprint list. The resulting growth is also reflected in the sprint burnout chart. Most teams find that about 1/3 of the way through the sprint, the burnout chart turns back around and continues downwards. sprint discovery of more work becomes a pattern, and some teams are initially shocked, but soon start to realise that this is what a normal sprint burnout chart looks like, starting with an upward curve and then really turning downwards. The pattern varies from team to team. If the team is very experienced and familiar with the codebase, the initial upward curve in the chart may be small, but if it is a young team taking on new territory, the curve may be quite pronounced.

3. TASK3

3.1 Terraform Enabling Infrastructure

Terraform is a tool for securely and efficiently generating, changing and versioning infrastructure, Terraform is an IT infrastructure automation tool with the slogan "Write, Plan, and create Infrastructure as Code". Specifically, we can use code to manage and maintain IT resources, for example for AWS, we can use it to create, modify and delete S3 Bucket, Lambda, EC2 instances, Kinesis, VPC and many more. And you can see the execution plan before it actually runs. Since the state is saved to a file, it is possible to view the resources offline. Terraform can manage both low-level modules such as compute instances, storage, networking, and high-level modules such as DNS entries, SaaS functionality, etc.

Terraform's main features are: 1) Infrastructure as Code (IaC). Using a high-level configuration syntax to describe the infrastructure, the data centre blueprint can be versioned, just like any other code. In addition, the infrastructure can be shared and reused; (ii) Execution Plans Terraform has a Planning step to generate execution plans. The execution plan shows what changes Terraform is going to make when it executes a Terraform apply, avoiding surprises when maintaining the infrastructure; (iii) Resource map. Terraform generates a resource map and then creates and modifies non-dependent resources in parallel. As a result, Terraform can efficiently build infrastructure; iv) Automate changes. Complex changes can be applied to the infrastructure with minimal human intervention, avoiding as many human errors as possible.

3.2 Provider

Terraform is designed as a multi-cloud infrastructure orchestration tool, unlike CloudFormation which is tied to the AWS platform, Terraform can orchestrate resources from various cloud platforms or other infrastructures at the same time, the way Terraform implements multi-cloud orchestration is through the Provider plugin mechanism. As shown in Figure 2 Terraform calls the plugin via RPC and the plugin code manipulates the remote resources by calling the SDK.

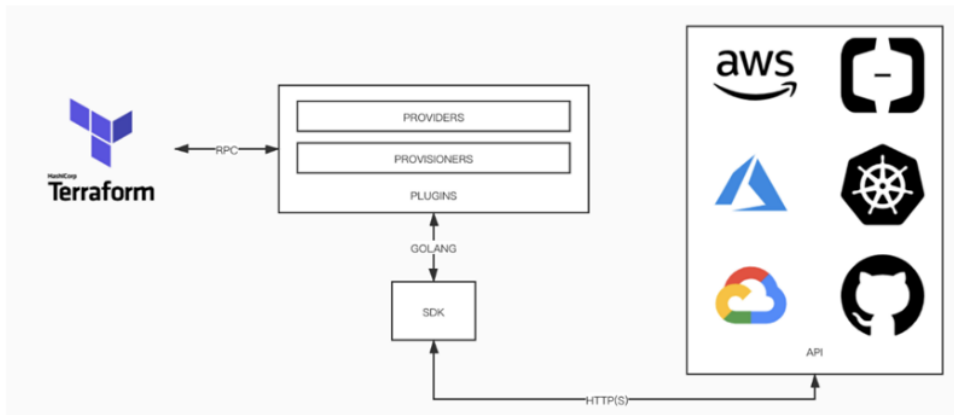


Figure 2. Plugin code manipulates remote resources by calling the SDK.

Terraform uses HashiCorp's own go-plugin library, and each Provider plugin is essentially a separate process that is called via rpc from the Terraform process. The writer of the Provider plugin defines the various data and resources according to the plugin framework defined by Terraform and implements the corresponding CRUD methods; in implementing these CRUD methods, either by calling the SDK provided by the target platform or by directly calling the Http(s) API to operate on the target platform.

Sometimes it can be very slow to download certain Providers, or there are many Terraform projects in the development environment and each project has its own folder of plugins which is very wasteful of disk, so we can use the plugin cache.

There are two ways to enable plugin caching.

The first method is to configure the environment variable `TF_PLUGIN_CACHE_DIR`:
`export TF_PLUGIN_CACHE_DIR="$HOME/.terraform.d/plugin-cache"`

The second method is to use the CLI configuration file, which is created under the `%APPDATA%` directory of the relevant user as "terraform.rc" for Windows and ".terraformrc" for MacOS and Linux users under the user's home. Configure the file as follows.

```
plugin_cache_dir = "$HOME/.terraform.d/plugin-cache"
```

When plugin caching is enabled, whenever the terraform init command is executed, the Terraform engine will first check if the desired plugin already exists in the cache folder, and if so, then copy the cached plugin to the .terraform folder in the current working directory. If the plugin does not exist then Terraform will still download the plugin as before and save it first in the plugins folder and later copy it from the plugins folder to the .terraform folder in the current working directory. To try to avoid the same copy of a plugin being saved multiple times, Terraform will use symbolic links rather than actually copying from the plugin cache directory to the working directory, as long as the operating system provides support. the Terraform

engine never actively deletes plugins from the cache folder, and the size of the cache folder may grow to be very large over time, at which point it will need to be cleaned up manually.

State Management

Once we have successfully executed a terraform apply and created the desired infrastructure, if we execute the terraform apply again, the new execution plan generated will not contain any changes and Terraform will remember the current state of the infrastructure and compare it to the desired state described by the code. On the second apply, an empty execution plan is generated because the current state is already consistent with the state described by the code.

Here, Terraform introduces a unique concept - state management - that is not present in configuration management tools such as Ansible or in the solutions of self-developed tools that invoke the SDK to operate the infrastructure. In simple terms, Terraform stores state information for each infrastructure change performed in a state file, which by default is stored in the terraform.tfstate file in the current working directory. For example, if we declare a data and a resource in our code.

```
data "ucloud_images" "default" {
  availability_zone = "cn-sh2-01"
  name_regex       = "^CentOS 6.5 64"
  image_type       = "base"
}
```

```
resource "ucloud_vpc" "vpc" {
  cidr_blocks = ["10.0.0.0/16"]
  name       = "my-vpc"
}
```

The data queried and the resource created are stored in a tfstate file in json format. Due to the existence of the tfstate file, no changes will be implemented if we apply again immediately after terraform apply, so what happens if we delete this tfstate file and then execute apply again? resources, so it will create all the resources described in the code again. To add to the problem, since the state information corresponding to the resources we created last time is deleted, we can no longer destroy and recycle these resources by executing terraform destroy, effectively creating a resource leak. This is why it is so important to keep this state file safe.

In addition, if we make some changes to Terraform's code that will cause the generated execution plan to change state, Terraform will copy a copy of the current tfstate file to terraform.tfstate.backup in the same path before actually executing the changes, to prevent any accidental tfstate corruption. To solve the problem of storing and sharing state files, Terraform introduces a remote state storage mechanism, known as Backend, which is an abstract remote storage interface and, like Provider, supports a number of different remote storage services.

There are two types of Terraform Remote Backend.
Standard: supports remote state storage and state locking

Enhanced: support for remote operations (plan, apply, etc. on remote servers) on top of the standard.

Currently the only enhanced Backend is the Terraform Cloud service.

A state lock is a global lock that can be added to a state file when a change is made to a tfstate, ensuring that only one change can be executed at a time. For example, the consul backend uses a .lock node to act as a lock, a .lockinfo node to describe the session information for the lock, and the tfstate file is stored in the path node defined by the backend; the s3 backend requires the user to pass in a Dynamodb table to store the lock information, and the tfstate file is stored in the s3 storage bucket. The backend named etcd corresponds to etcd v2, which does not support state locks; etcdv3 provides support for state locks, etc etc. I will then use consul as an example to demonstrate the backend mechanism.

3.3 General scenarios

In the following schema, we open the green line through Terraform to create all resources under the managed account on the right and automate them. This of course includes compute resources. We then roll out an Ansible configuration management host as the deployment server, and then roll out the corresponding systems under the target account. As long as there is a standard configuration method, we can achieve to automated creation. As for exactly what monitoring platform, test platform, container platform, it's all possible.

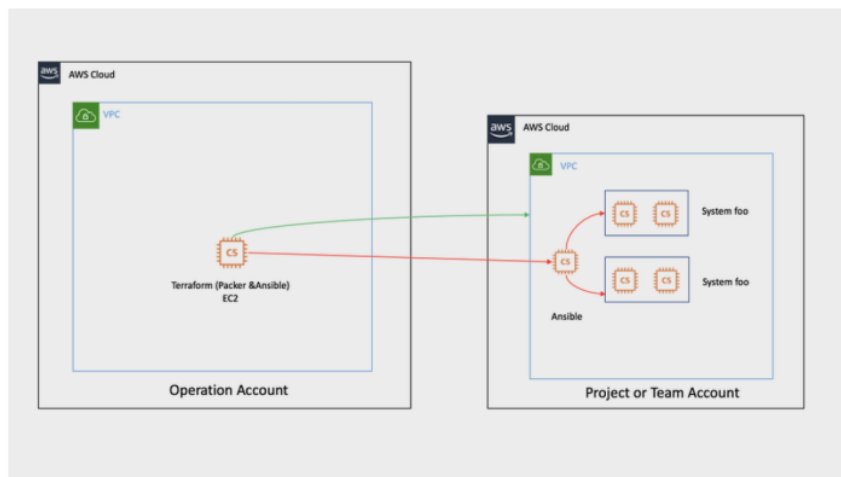


Figure 3. General scenarios

3.4 Enabling customised rapid deployment of Openshift 3 on AWS

Below is a diagram of a typical openshift deployment architecture on AWS. The hosts are already operating on this host, thus performing the configuration and management of the entire cluster environment.

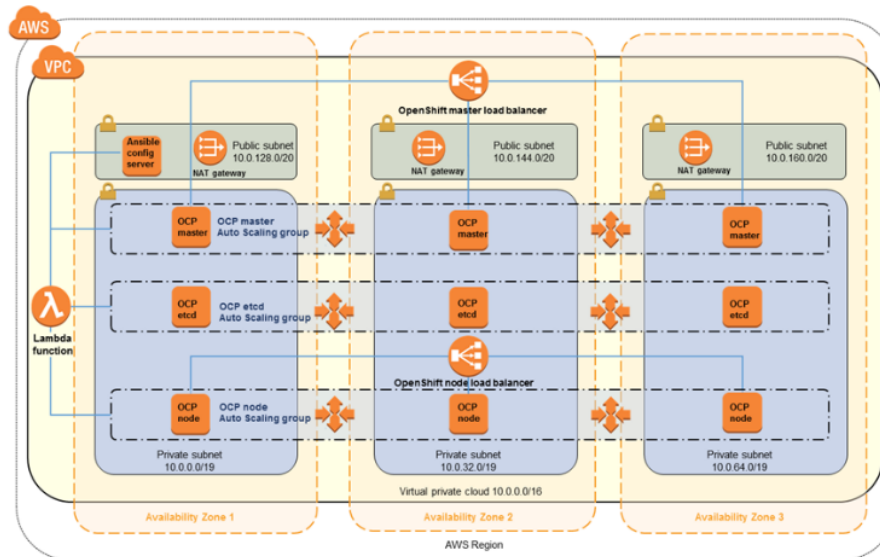


Figure 4. openshift deployment architecture diagram on AWS

Similarly, there are many tools for deploying native Kubernetes on AWS, e.g. kops, to quickly build up an environment on AWS. But in terms of plugin tuning, system configuration and tuning, there is basically no entry point at all for you to fine-tune the entire cluster architecture yourself. With our terraform + ansible combination, you can control all levels from the underlying resources to the system OS.

References

- [1] Xu Wei, Wang Haoyu, Xie Meng et al. Research on Scrum agile software development combined with CMMI [J]. Computer Technology and Development, 2014(9):89-92.
- [2] Yin M, Jiang D, Ma J et al. Collaborative game study of distributed agile development teams in different locations[J]. Northwestern Polytechnic University Journal of Northwestern Polytechnic University, 2016, 34(4):714-719.
- [3] Zeng Rong. Scrum Agile Practices and Benefits [D]. Beijing: Beijing University of Posts and Telecommunications, 2009.
- [4] He Jing. Application of Scrum agile method in software project management[J]. Digital Technology and Applications, 2021, 39(03):87-89.
- [5] Yan Shuai, Xu Pengxiang. A project management approach based on waterfall model combined with agile development[J]. Electronic Technology and Software Engineering Electronic Technology and Software Engineering, 2013,(18):67.
- [6] Zeng Xiaoxue, He Dongmei, Xie Hua. Study on the application of agile project management method[J]. Petroleum Planning and Design, 2018, 29(02):44-47.

FINAL GRADE

80/100

GENERAL COMMENTS

Instructor

Agile (40): 30

Story Est (20): 17

SwDevOp Tool (40): 33

Feedback

=====

Some of the Agile discussion is a little confusing, but you have clearly addressed most of the important concepts. It would have helped if you had defined company A, B, C and D near the beginning to emphasise their different characteristics and requirements. Also you should have provided more examples of the process

The story estimation process is discussed in good detail, I would like to have seen a more illustrative example.

Your terraform discussion correctly identifies the use of a state to allow it to deploy needed changes. This declarative approach is the crucial concept and I think you could have expanded on this idea. However, you have provided a good overall discussion of the tool. The scenario did not really bring out the benefits of using Terraform.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16



Comment 1

This is the crucial concept, and I would have liked you to expand upon what this means to the user.

PAGE 17

PAGE 18

PAGE 19
