Cloudeploy 2.0
Requirement Analysis

2016-07-20

Catalogue

Cloudeploy 2.0 Requirement Analysis

1. Introduction

Cloudeploy 2.0 is an open-source platform for DevOps (Development and Operations). Its functions include deployment, configuration, system scaling, fault-tolerance and so on. With the services of Cloudeploy 2.0, one can easily manage cloud based applications and middleware in a distributed cluster through web browser instead of CLI (command-line interface).

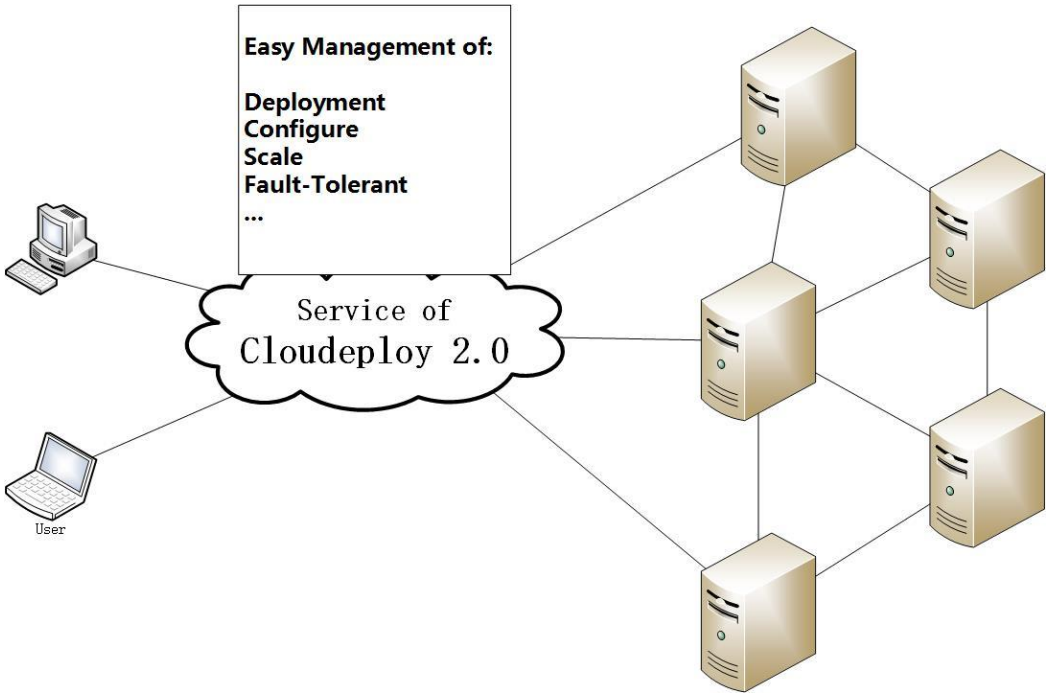

Figure 1.1 System Schematic

The intention is that when developers build up environments for specific applications, such as a Java Web which may need middleware components like tomcat, MySQL, etc. Easy and continuous deployment tools will help developers focus on development rather than building environments and configuring middleware components with huge efforts. Cloudeploy 2.0 is just to make these work automatic and easy going.

2. Feasibility Analysis

2.1. Present Situation Analysis

Besides CLI, there are some ways to semi-automatically manage computers, such as shell scripts, CMT (configure management tools), containers and so on. But restricted by objective conditions, these ways require domain specific knowledge, making it is hard for users. And when deploying a middleware component, one must know clearly about the system installation topology, the execution scripts and the domain specific language (DSL). And then a set of actions should be performed step by step under some instructions. Therefore, a graphic based, model-driven DevOps deployment platform is urgently needed.

2.2. Market Value Analysis

According to IDC (2012), more than 50% of large companies were going to move their infrastructure of computing to cloud. A survey taken by Right Scale shows that more than 70% of companies use automatic tools to manage computer. Further, much more small companies are using cloud computing platform such as AWS, Azure, Aliyun, which make cloud based applications widespread. As a result, management of cloud based applications such as deployment is playing an important role in industry of cloud computing.

2.3. Feasibility and Technology Analysis

With the development of cloud computing there are lots kinds of technology for management of computer and applications. For example, skill of virtualization can be used in resource distributions; CMT tools may be used to install and update software in target machines or virtual machines; Docker is a kind of container which holds a specific process of software component.

These technologies are still raw and time consuming for beginners. But we can do some further work based on them to simplify DevOps process.

3. Function Requirement Analysis

As a DevOps tool that support continuous deployment and configuration for cloud based applications, the architecture of Cloudeploy 2.0 consists of two parts: web server and agent client on computers of cluster.

On one hand, web server needs to provide the following functions:

- Create deployment task
- Configure parameters of applications
- Storage server for configuration templates
- Auto Scaling by adding components
- Components health checking

On the other hand, agent client should has the following functions:

- Execute deployment and configuration scripts.
- Dynamically generate configure files
- Generate healthy report of applications

Functions of Web Server

3.1. Create deployment task

A deployment task is an orchestration of a specific applications. Normally, an application consists of several software components, as is showed in figure 3.1. A java Web application may require Nginx as load balancer, tomcat as servlet container, MySQL as database, and java as runtime machine. In this case, we can summarize the tasks as follows.

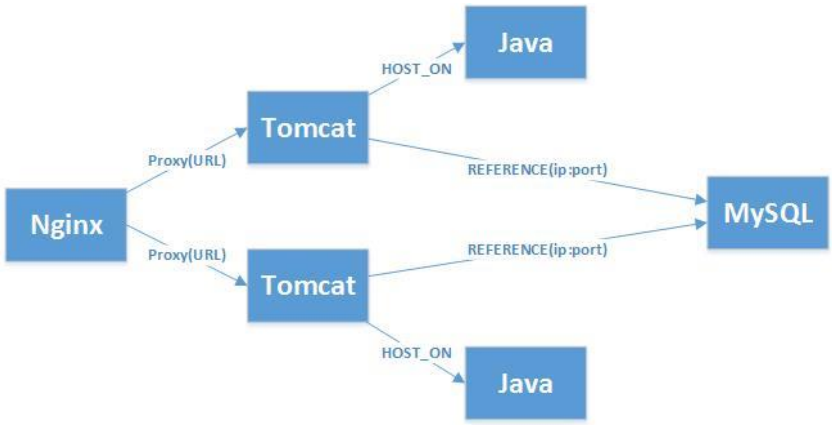

Figure 3.1 Components in Java Web application

- **Component type and instances number**

  An application consists of several components so we need to point out which component and how many instances of it. In Figure 3.1, the components and numbers can be concluded by: (Nginx, 1), (Tomcat, 2),(Java, 2),(MySQL, 1).

- **Type of operation**

  Since we know each component type, it still needs to define what operation that would be taken. The operations includes **install**, **start**, **stop** , **delete**, and **restart** may be made up by **stop** and **start**. Then if we want scaling applications by adding or removing components and component instances, it can be implemented by performing ***start*** or ***stop*** operations first.

- **Service topology and action sequence.**

  Some services provided by software component are based on the other services, for example tomcat based on java. This implies an order of installations that we should comply with.

- **Parameters relations**

  Some parameters of components are consistent with one another. For example, the parameters of “port”, ”username”, “password” of MySQL may be used by Tomcat for connection. And “encoding”, “engine type” will be not used by another component. So we need to distinguish these parameters and figure out their relations.

With these tasks described above, we need a model to make the task specifications. Further, to make easy to use, a graphical environment should be provided to modeling the process of application deployment.

3.2. Configure parameters of applications

There may be reconfigurations of a deployed application. So we should dynamically generate new configure files to replace the old one with a pre-defined template and then restart the component to apply the changes. To avoid the exceptions during a reconfiguration, a transaction mechanism is required.

It worth to note that there may be some other components relying on the changed one. As a result some cascading actions must be taken after the change.

3.3. Storage server for configure templates

When generating configuration files or task files for software components, we should get the template files first. Therefor, a storage server is required, and whose functions must include,

- **Upload files.**

  User edit templates offline, and upload it to Cloudeploy 2.0 for further usage.

- **Download files.**

  The files stored can be easily accessed by agent clients through RESTful URL.

- **Duplicate filter**

  If there is an exist file, then, it will prevent repeating action of uploading

3.4. Auto Scaling

When the request load varies, the number of instances of component may be adjusted. For example, in figure 3.1, we can add another tomcat instance for the sake of heavy load of requests to web server. When with fewer requests, some instances can be removed to saving computing resources.

Note that there may be some cascading actions after the change, for example, rebuilding the relation from new tomcat instance to MySQL, and the relation from Nginx to the new tomcat instance.

3.5. Components health checking

After deploying the application, we need to know the state of its components, such as if they are running or stopped.

- **Define command of check.**

  It has different meaning of "healthy" for the different components. So user can define specific command for healthy check. For example check command of MySQL is to confirm the process state of running, for tomcat, there may be a http request to try get correct response.

- **Execute check command**

  To check the state of components, we need to send the command to target agent with interval time setting. There will be a heartbeat mechanism through agent cluster.

- **Show result reports**

  To report the state of components, a warning of "error" components will be shown on Cloudeploy 2.0 dashboard. The different state may be shown in different colors.

Functions of agent client

Agent Clients are hosts of applications. The tasks of deployments or configurations will finally be applied on agent computer clusters. The agent may be machines, virtual machines or containers. No matter what kind of hosts, the following functions must be taken account of.

3.6. **Execute deployment and configuration scripts.**

- Agents fetch task events from dashboard, and get task scripts from storage server.
- Execute scripts on hosts, and get the result of execution.
- Return result to dashboard in form of given format.

3.7. Dynamically generate configuration files

- Agents fetch templates from storage server.
- Get parameters from dashboard server and generate configuration files for specific components.
- Put the configuration files in proper path of agents, and restart components.

3.8. Generate healthy report of applications

- Agents get check command from dashboard and execute command periodically.
- Generate reports of execution result, and send to dashboard.