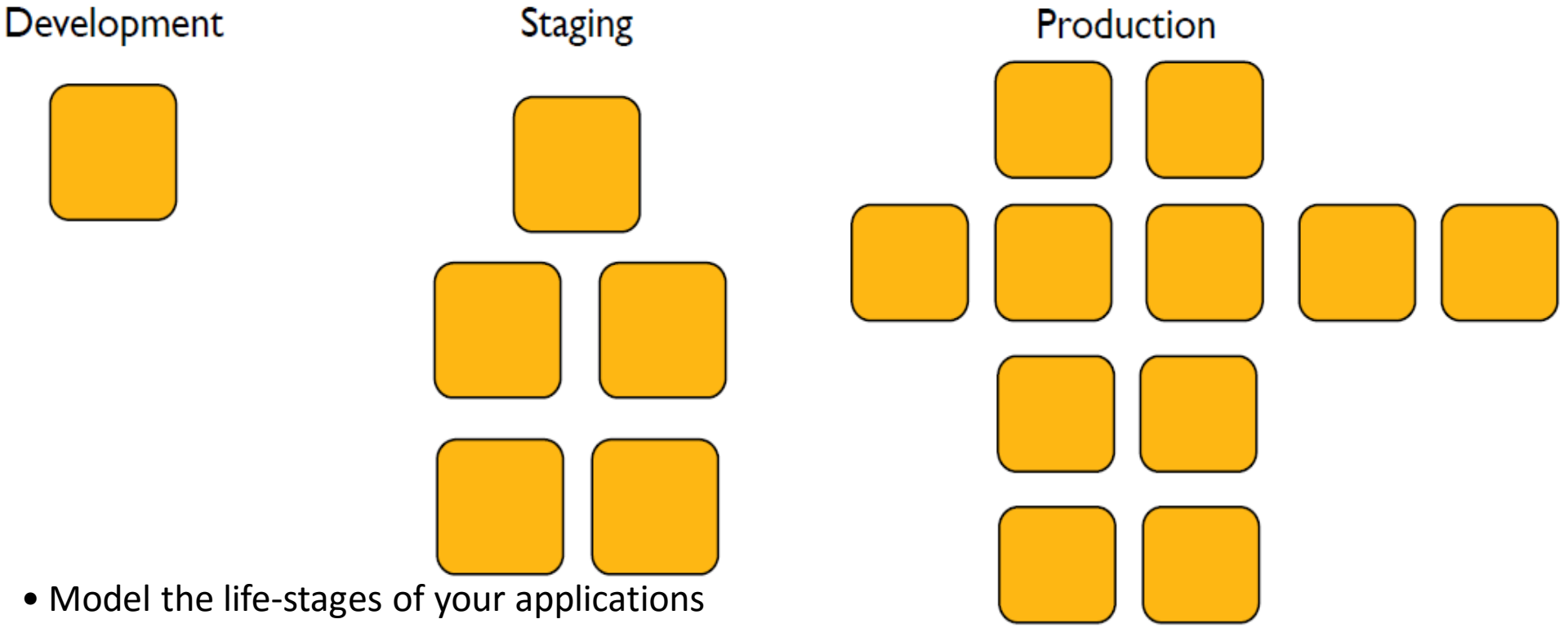
Chef Environment

- An environment is simply a designation meant to help an administrator know what stage of the production process a server is a part of. Each server can be part of exactly one environment.
- Environments that coincide with your actual product life-cycle make the most sense. If you run your code through testing, staging, and production, you should have environments to match.

Environments



- Model the life-stages of your applications
- Every Organization starts with a single environment
- Environments to reflect your patterns and workflow
    - Development
    - Test
    - Staging
    - Production
    - etc.

Environments Define Policy

Environments may include data attributes necessary for configuring your infrastructure

- The URL of your payment service's API
- The location of your package repository
- The version of the Chef configuration files that should be used

- By default, an environment called "_default" is created.
- Each node will be placed into this environment unless another environment is specified.
- Environments can be created to tag a server as part of a process group.

example

- For instance, one environment may be called "testing" and another may be called "production". Since you don't want any code that is still in testing on your production machines, each machine can only be in one environment. You can then have one configuration for machines in your testing environment, and a completely different configuration for computers in production.
- In the above example given in roles, you could specify that in your testing environment, the web and database server roles will be on a single machine. In your production environment, these roles should be tackled by individual servers.

Set Chef Environment in Roles

- For instance, if a node is in the "production" environment, you could want to run a special recipe in your "nginx" cookbook to bring that server up to production policy requirements. You could also have a recipe in the nginx cookbook meant to configure special changes for testing servers.
- Assuming that these two recipes are called "config*prod" and "config*test" respectively, we could create some environmental specific run lists like this:

Attribute Types

There are two types of attributes that can be used with environments:

Attribute Type	Description
`default`	A `default` attribute is automatically reset at the start of every chef-client run and has the lowest attribute precedence. Use `default` attributes as often as possible in cookbooks.
`override`	An `override` attribute is automatically reset at the start of every chef-client run and has a higher attribute precedence than `default`, `force_default`, and `normal` attributes. An `override` attribute is most often specified in a recipe, but can be specified in an attribute file, for a role, and/or for an environment. A cookbook should be authored so that it uses `override` attributes only when required.

Attribute precedence, when viewed as a table:

	Attribute Files	Node / Recipe	Environment	Role
default	1	2	3	4
force_default	5	6		
normal	7	8		
override	9	10	12	11
force_override	13	14		
automatic		15		

knife role create test

```
{
  "name": "test",
  "description": "",
  "json_class": "Chef::Role",
  "default_attributes": {
  },
  "override_attributes": {
  },
  "chef_type": "role",
  "run_list": [

  ],
  "env_run_lists": {
  }
}
```

```
{
  "name": "web_server",
  "description": "A role to configure our front-line web servers",
  "json_class": "Chef::Role",
  "default_attributes": {
    "nginx": {
      "log_location": "/var/log/nginx.log"
    }
  },
  "override_attributes": {
    "nginx": {
      "gzip": "on"
    }
  },
  "chef_type": "role",
  "run_list": [
    "recipe[apt]",
    "recipe[nginx]"
  ],
  "env_run_lists": {
    "production": [
      "recipe[nginx::config_prod]"
    ],
    "testing": [
      "recipe[nginx::config_test]"
    ]
  }
}
```

In the above example, we have specified that if the node is part of the production environment, it should run the "config*prod" recipe within the "nginx" cookbook.*

*However, if the node is in the testing environment, it will run the "config*test" recipe. If a node is in a different environment, then the default run_list will be applied.

- Similarly, we can specify default and override attributes. You should be familiar with default attributes at this point. In our role, we can set default attributes which can override any of the default attributes set anywhere else.
- We can also set override attributes, which have a higher precedence than many other attribute declarations. We can use this to try to force nodes that are assigned this role to behave in a certain way.

How To Use Environments

> knife environment create development

```
{
  "name": "development",
  "description": "The master development branch",
  "cookbook_versions": {
    "nginx": "<= 1.1.0",
    "apt": "= 0.0.1"
  },
  "json_class": "Cheff:Environment",
  "chef_type": "environment",
  "default_attributes": {
  },
  "override_attributes": {
    "nginx": {
      "listen": [
        "80",
        "443"
      ]
    },
    "mysql": {
      "root_pass": "root"
    }
  }
}
```

Setting Environments in Nodes

For instance, to edit a node called `client1`, we could type this:

```
knife node edit client1
```

This will open up a JSON formatted file with the current node parameters:

```
{
  "name": "client1",
  "chef_environment": "_default",
  "normal": {
    "tags": [

    ]
  },
  "run_list": [
    "role[web_server]"
  ]
}
```

As you can see, the `chef_environment` is set to `_default` originally. We can simply modify that value to put the node into a new environment.

Reference

- https://docs.chef.io/environments.html
- https://www.digitalocean.com/community/tutorials/how-to-use-roles-and-environments-in-chef-to-control-server-configurations