Application software configuration using Heat

Steve Baker
Senior Software Engineer, Red Hat

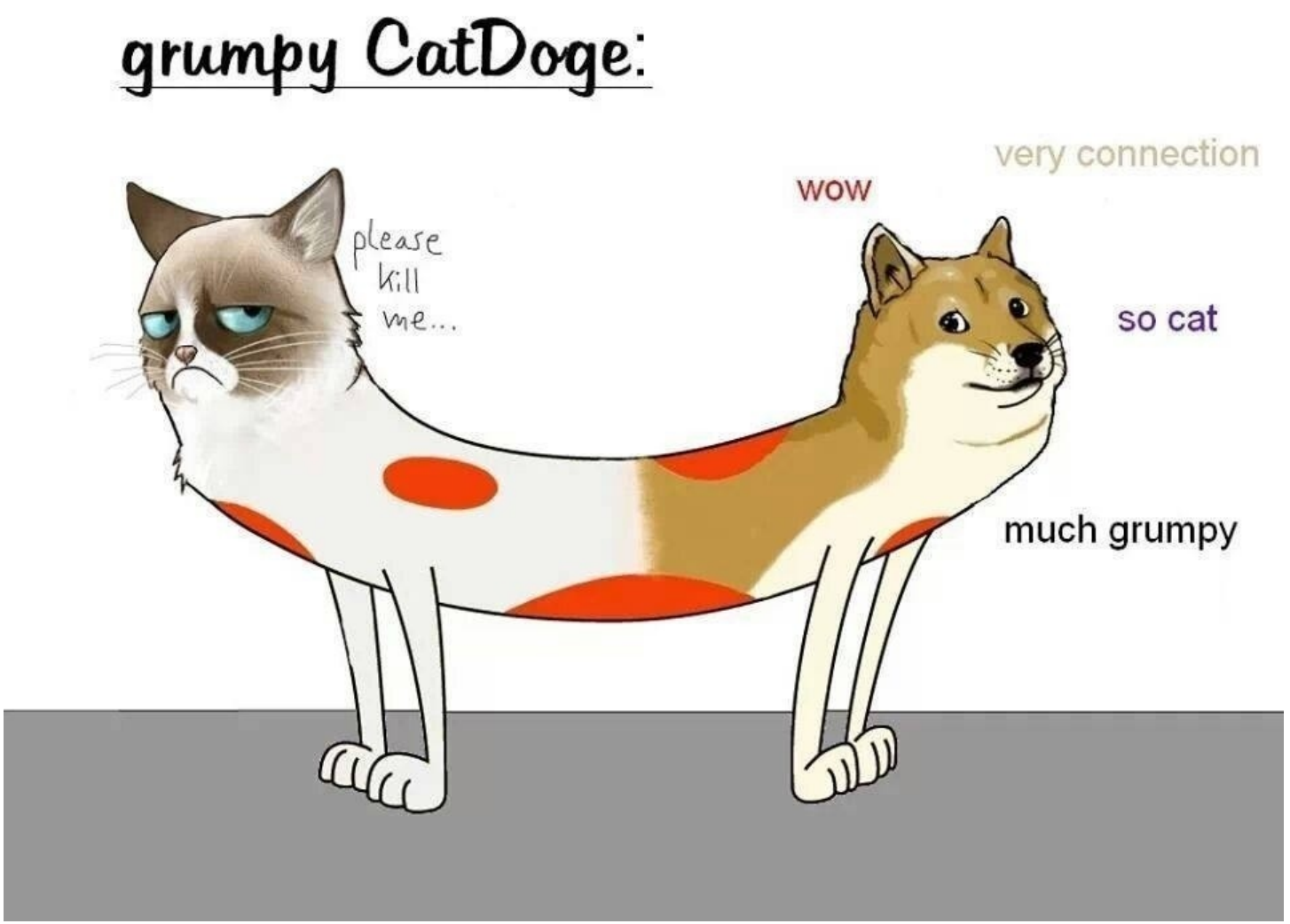
sbaker@redhat.com
irc stevebaker #heat

Application software configuration using Heat

- Configuration vs Orchestration
- New heat software config and deployment resources
- Integrating configuration tools

Software Configuration <> Orchestration

Separation of
concerns is important



Choosing an abstraction involves compromise

CloudFormation cfn-init example

```
"Resources" : {
  "WikiDatabase": {
    "Type": "AWS::EC2::Instance",
    "Metadata" : {
      "AWS::CloudFormation::Init" : {
        "config" : {
          "packages" : {
            "yum" : {
              "mysql"        : [],
              "mysql-server" : [],
              "httpd"        : [],
              "wordpress"    : []
            }
          },
          "services" : {
            "systemd" : {
              "mysqld"   : { "enabled" : "true", "ensureRunning" : "true" },
              "httpd"    : { "enabled" : "true", "ensureRunning" : "true" }
            }
          }
        }
      }
    },
```

CloudFormation cfn-init example

```
"Properties": {
  "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
    "#!/bin/bash -v\n",
    "/opt/aws/bin/cfn-init\n",
    "# Setup MySQL root password and create a user\n",
    "mysqladmin -u root password '", { "Ref" : "DBRootPassword" }, "'\n",
    "cat << EOF | mysql -u root --password='", { "Ref" : "DBRootPassword" }, "'\n",
    "CREATE DATABASE ", { "Ref" : "DBName" }, ";\n",
    "GRANT ALL PRIVILEGES ON ", { "Ref" : "DBName" }, ".* TO \"", { "Ref" : "DBUs
    "IDENTIFIED BY \"", { "Ref" : "DBPassword" }, "\";\n",
    "FLUSH PRIVILEGES;\n",
    "EXIT\n",
    "EOF\n",
    "sed -i \"/Deny from All/d\" /etc/httpd/conf.d/wordpress.conf\n",
    "sed -i \"s/Require local/Require all granted/\" /etc/httpd/conf.d/wordpress.con
    "sed --in-place --e s/database_name_here/", { "Ref" : "DBName" }, "/ --e s/usern
    "systemctl restart httpd.service\n",
    "firewall-cmd --add-service=http\n",
    "firewall-cmd --permanent --add-service=http\n"
  ]]}}
```

Both have roles to play
in the stack

Configuration resource

- API backed store of configuration data
- Stores configuration script
- Defines inputs and outputs schema
- Tool specific options
- Are immutable and can be passed by referenced

Boot configuration with cloud-init

```
one_init:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
      - path: /tmp/one
        content: "The one is bar"

two_init:
  type: OS::Heat::SoftwareConfig
  properties:
    config: |
      #!/bin/sh
      echo "The two is bar" > /tmp/two
```

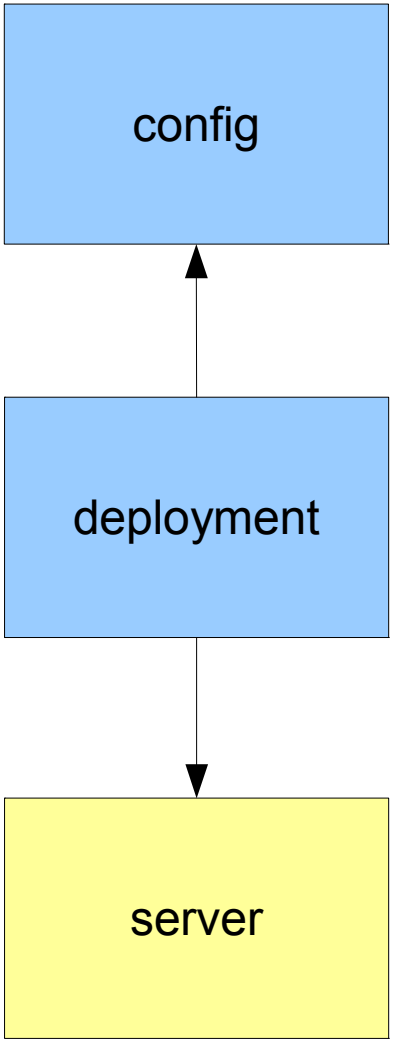
```
server_init:
  type: OS::Heat::MultipartMime
  properties:
    parts:
    - config: {get_resource: one_init}
    - config: {get_resource: two_init}

server:
  type: OS::Nova::Server
  properties:
    image: {get_param: image}
    flavor: {get_param: flavor}
    key_name: {get_param: key_name}
    user_data_format: RAW
    user_data:
      get_resource: server_init
```

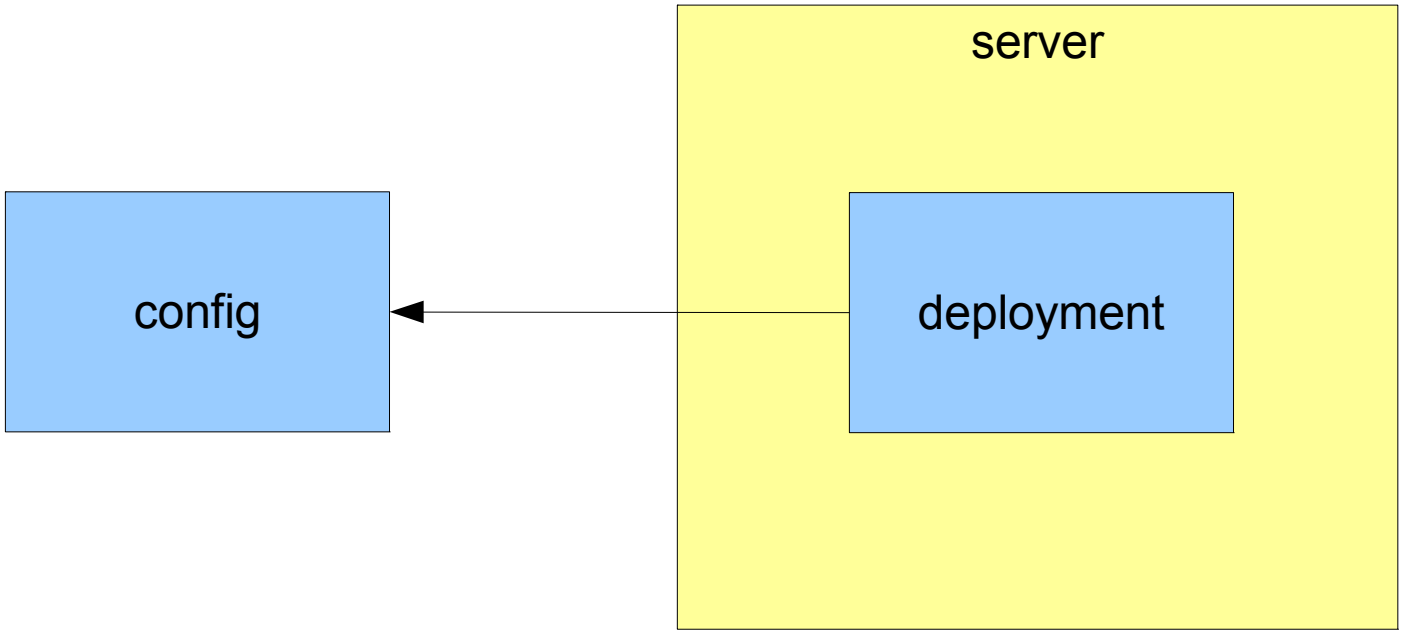
Deployment resources

- Maps one config resource to one server resource
- Allows assignment of server-specific input values
- Remains in-progress until receiving completed signal
- Stores outputs for other resources to consume as resource attributes
- Can deploy on any heat action, not just CREATE, UPDATE
- Stores additional outputs from hook invocation
  - stdin, stdout, status_code

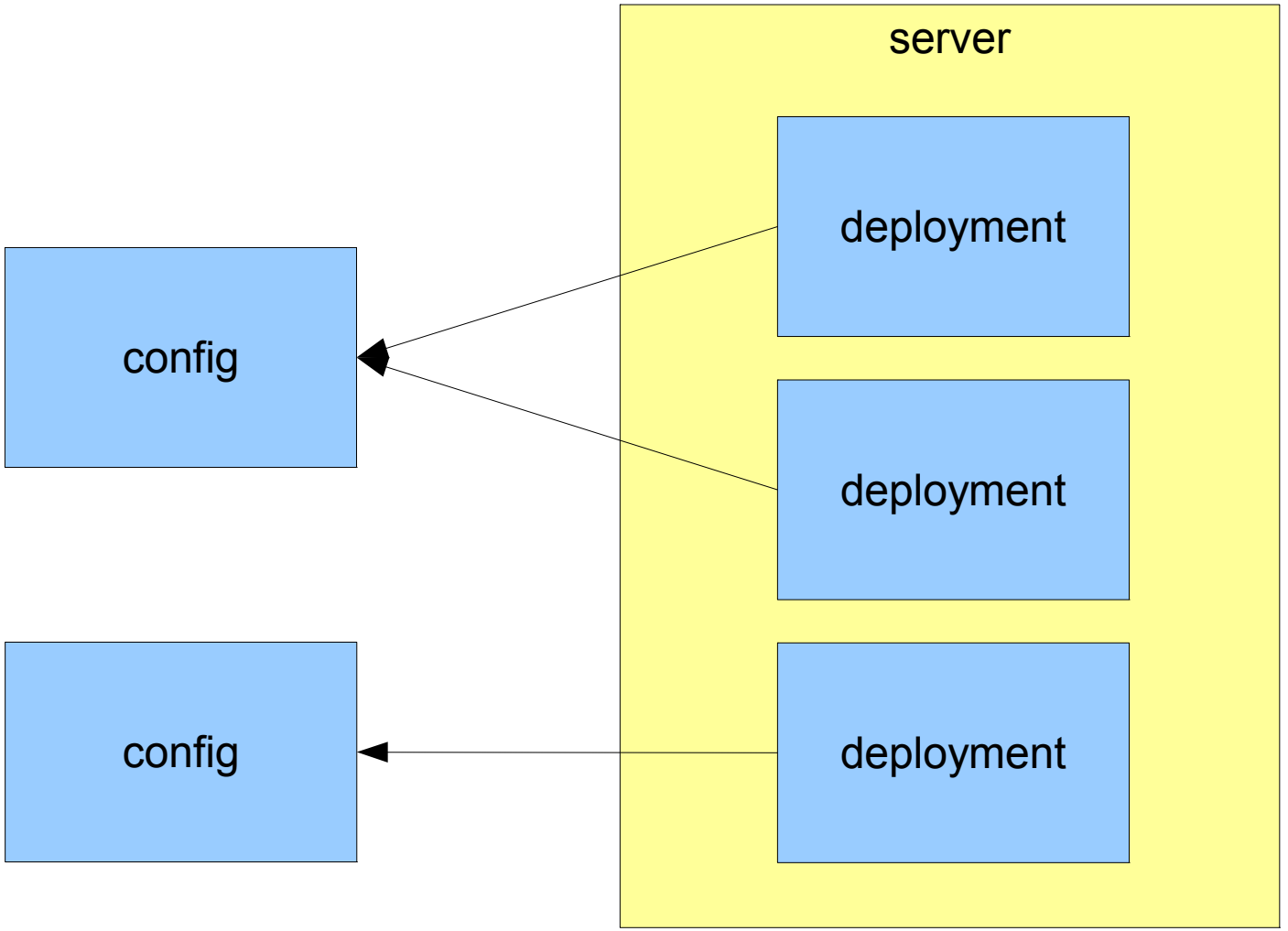
Deployment illustrated



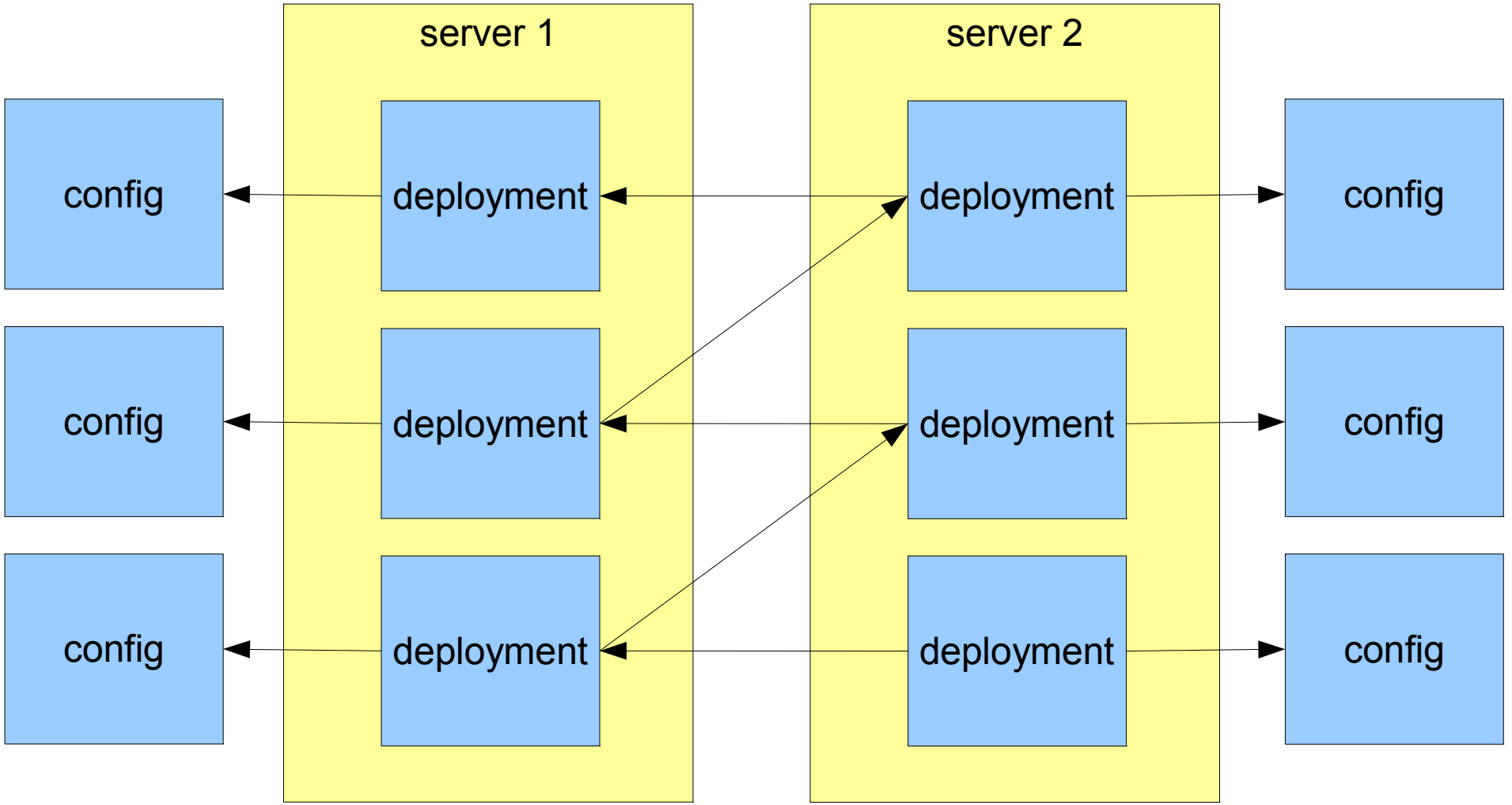
Deployment illustrated



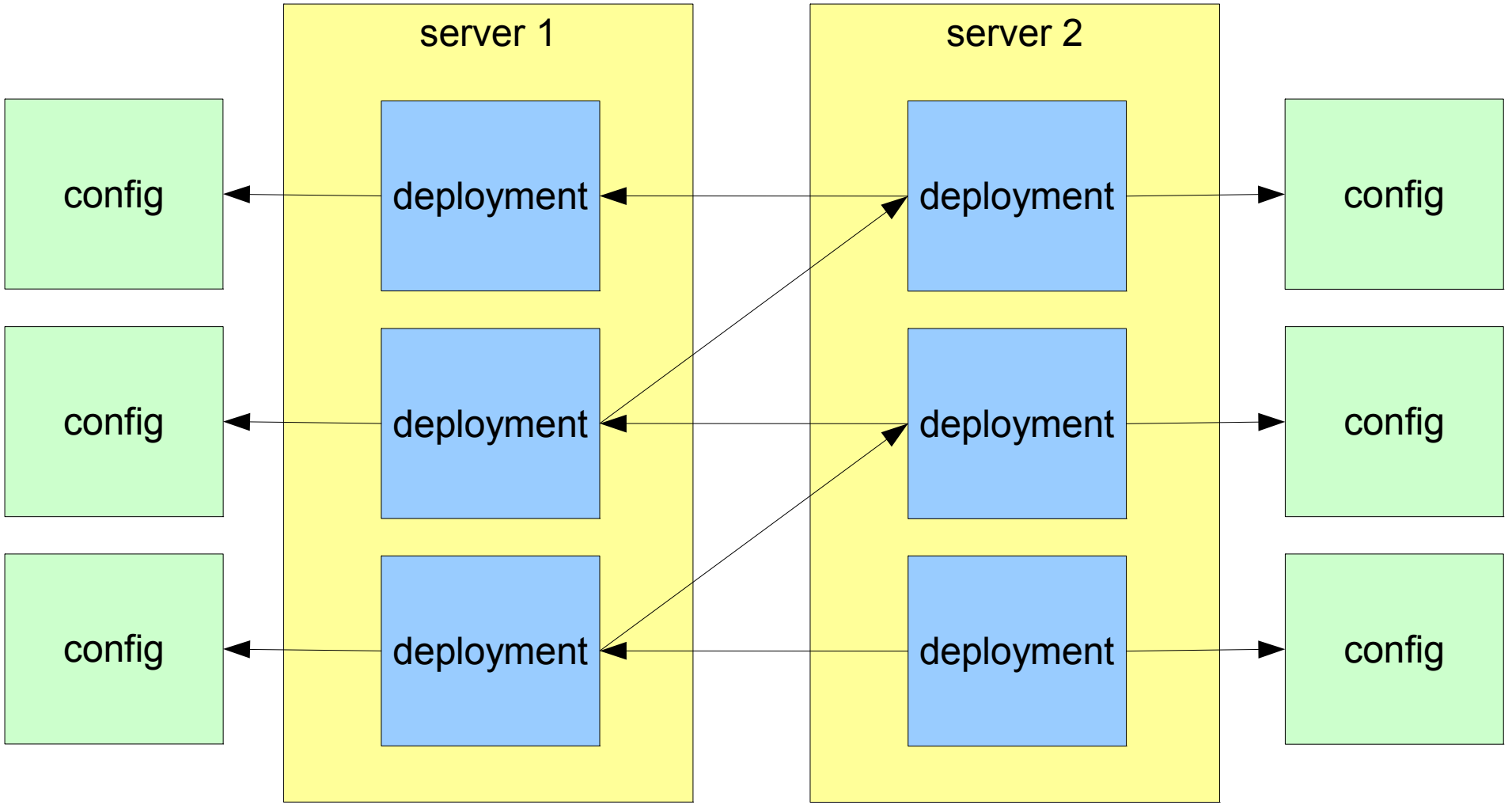
Deployments illustrated



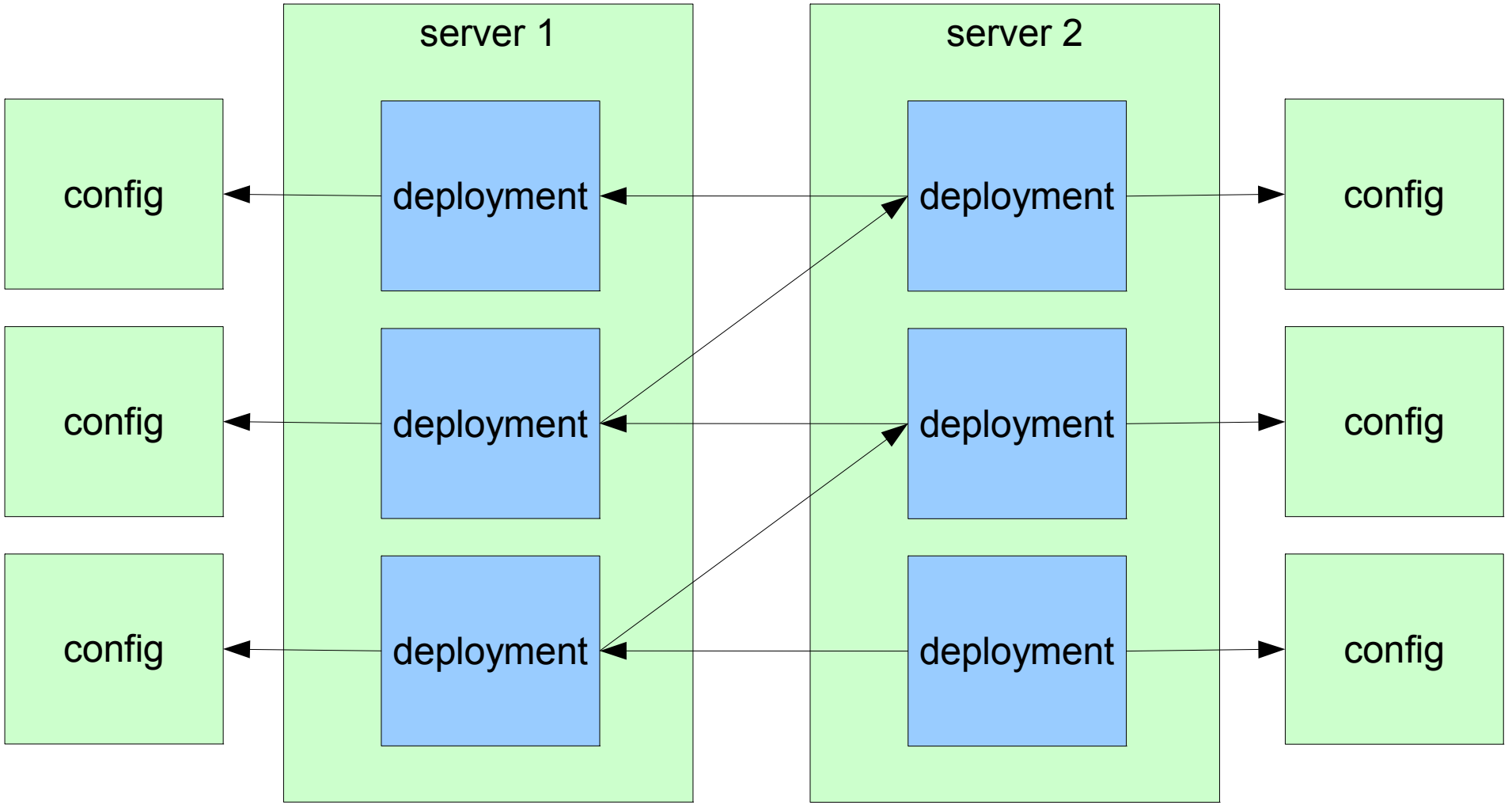
Deployments illustrated



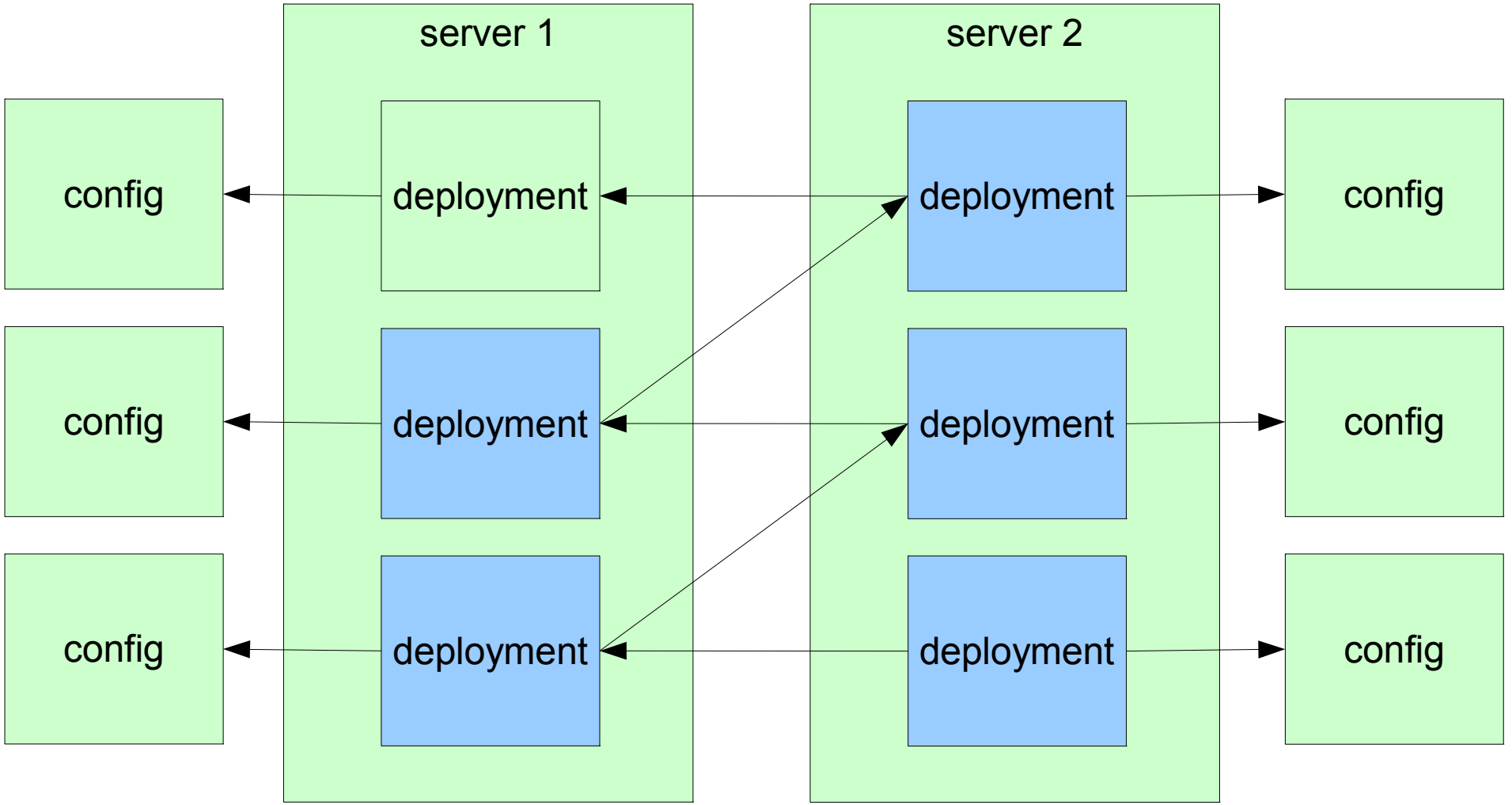
Deployments illustrated



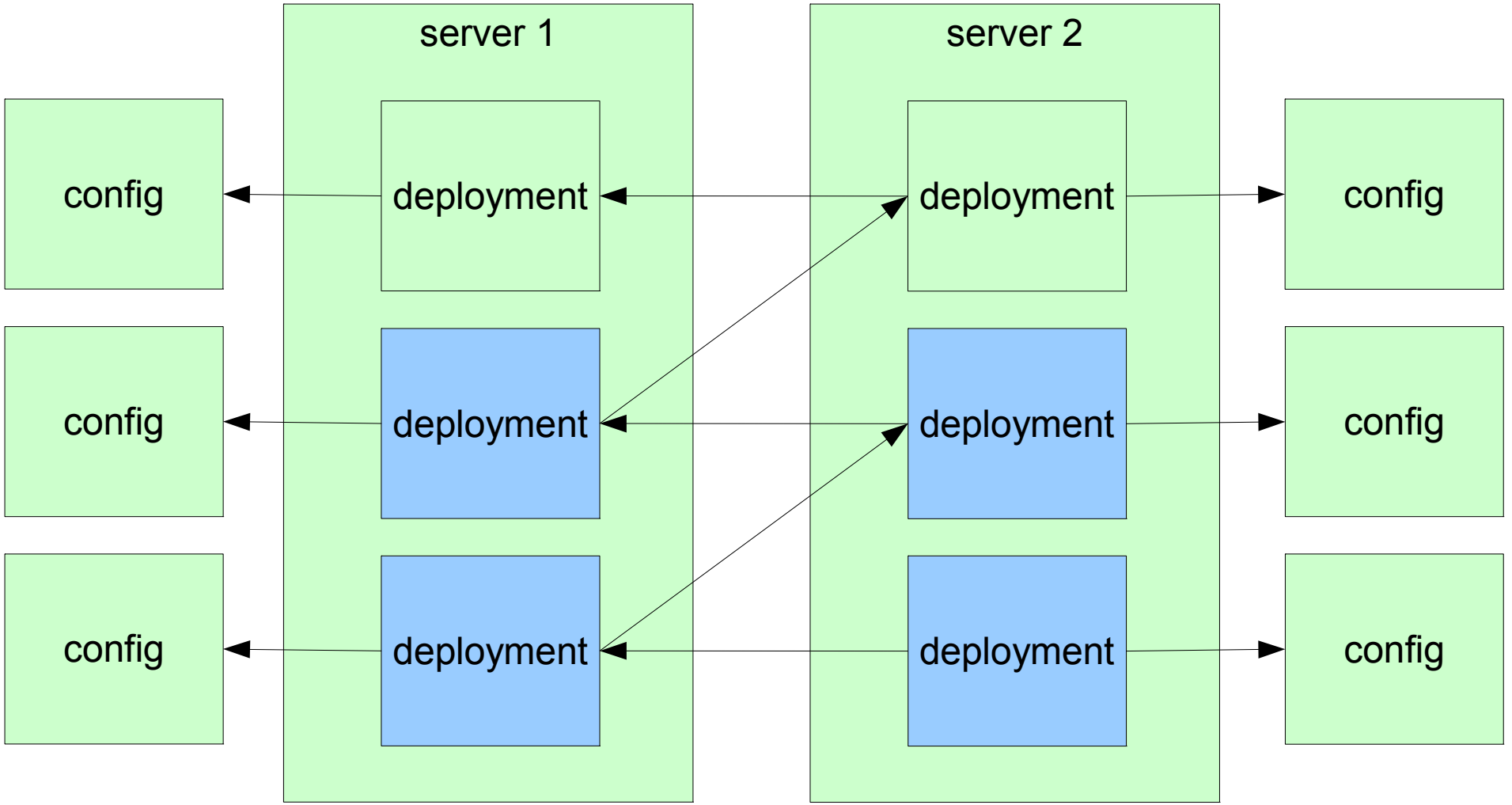
Deployments illustrated



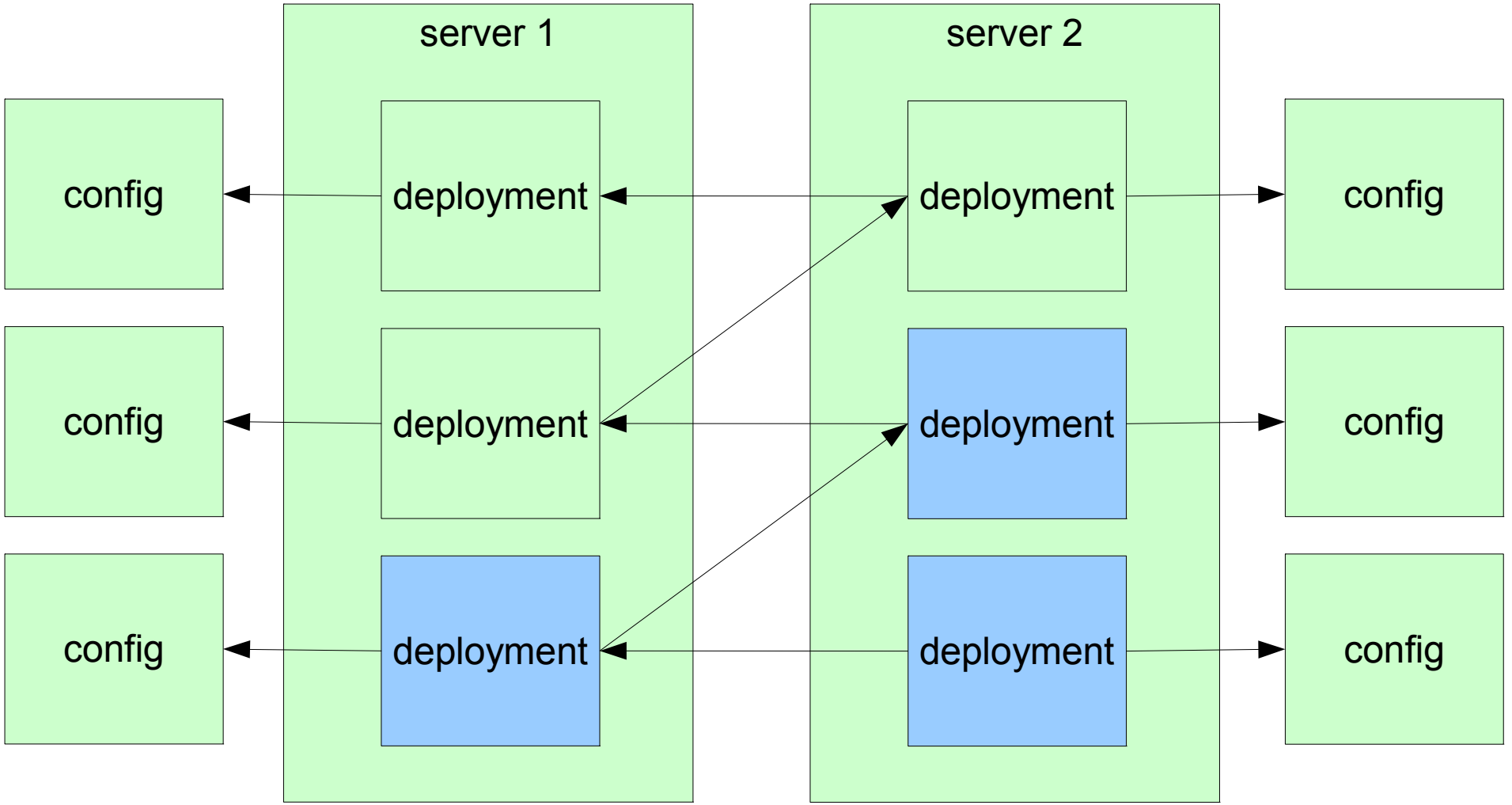
Deployments illustrated



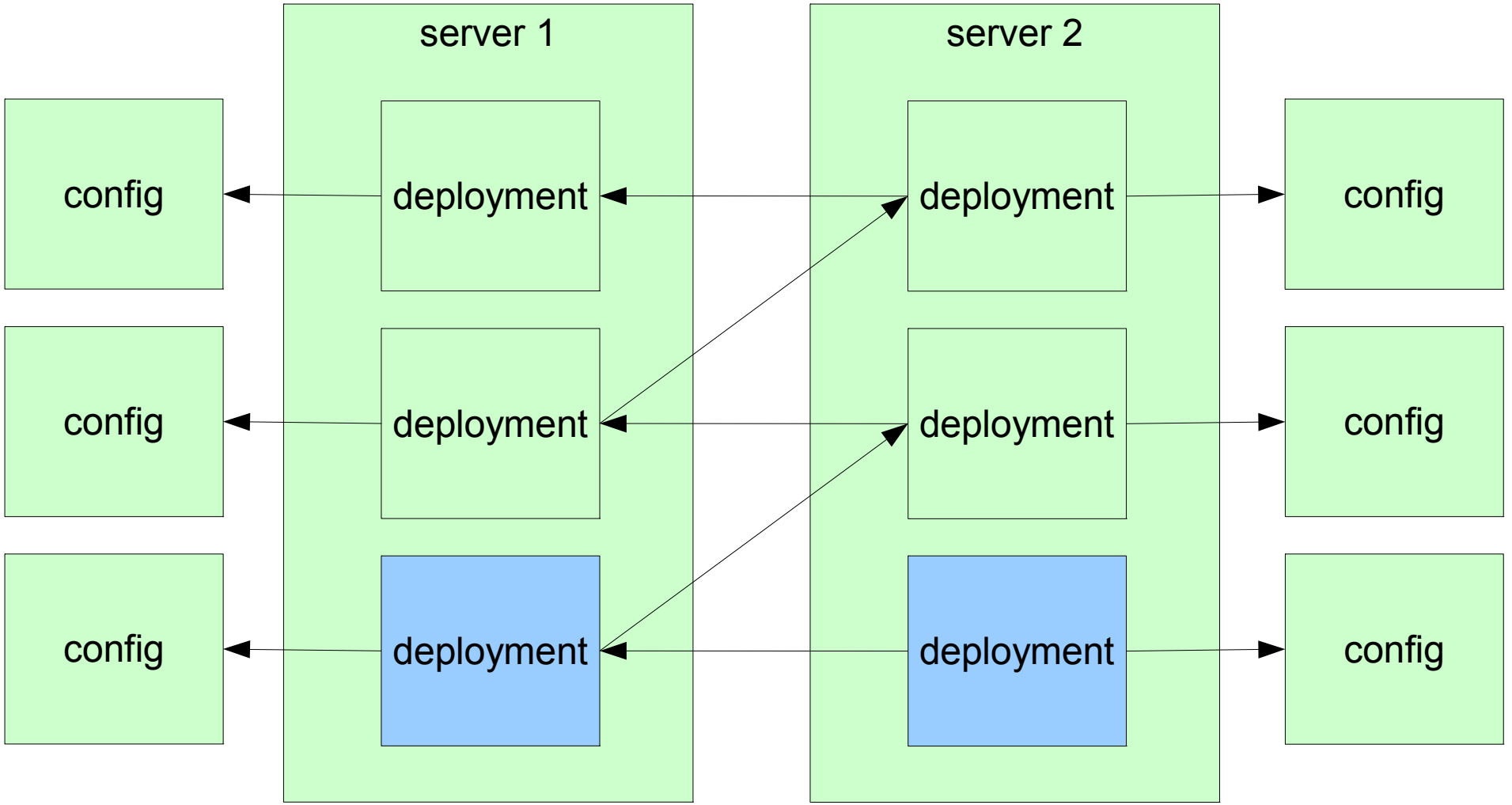
Deployments illustrated



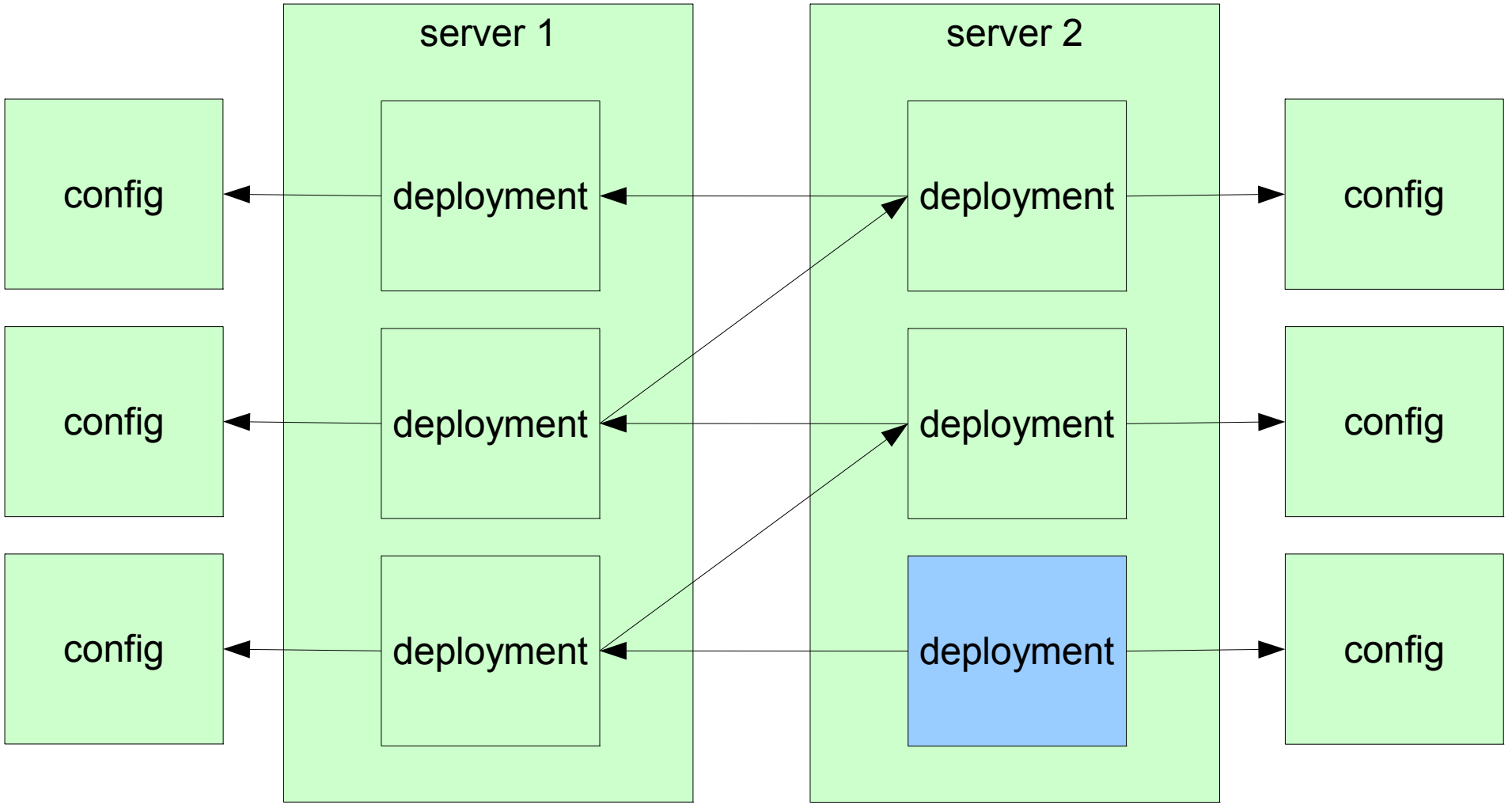
Deployments illustrated



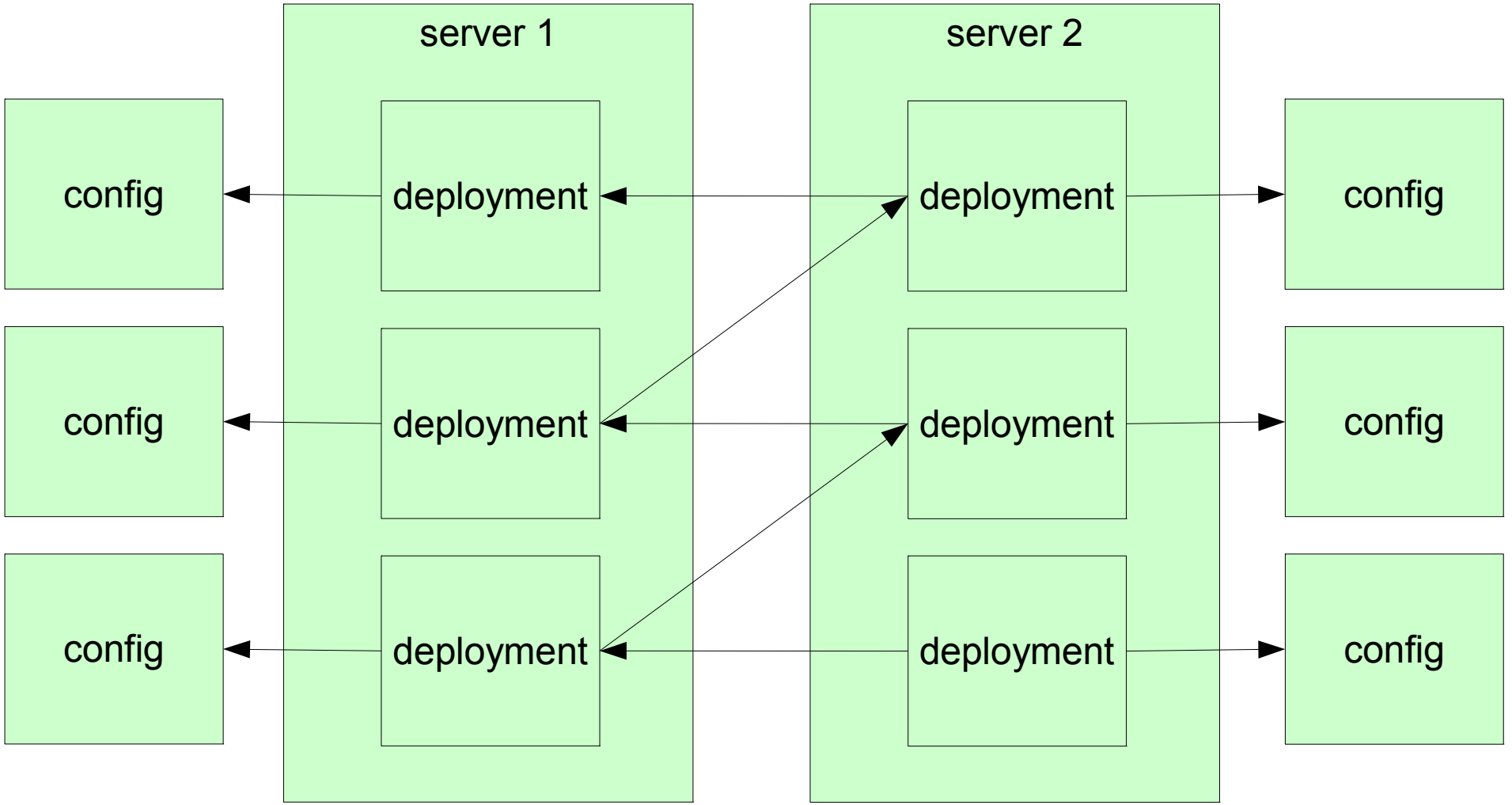
Deployments illustrated



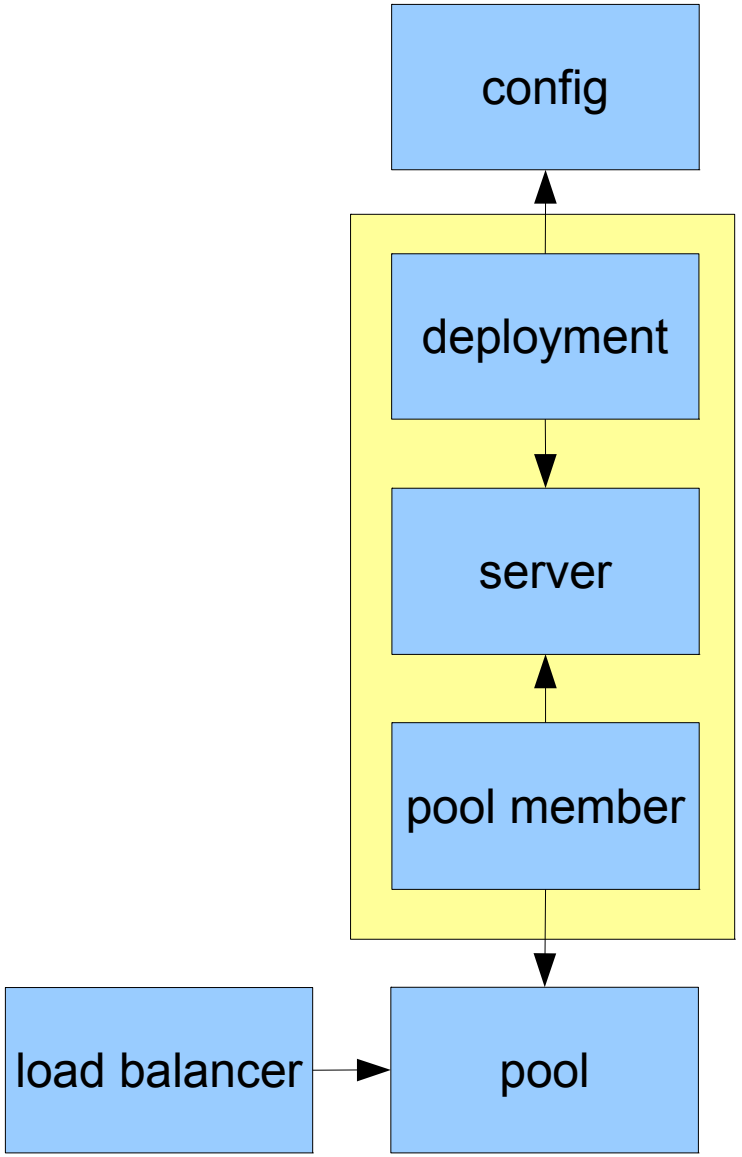
Deployments illustrated



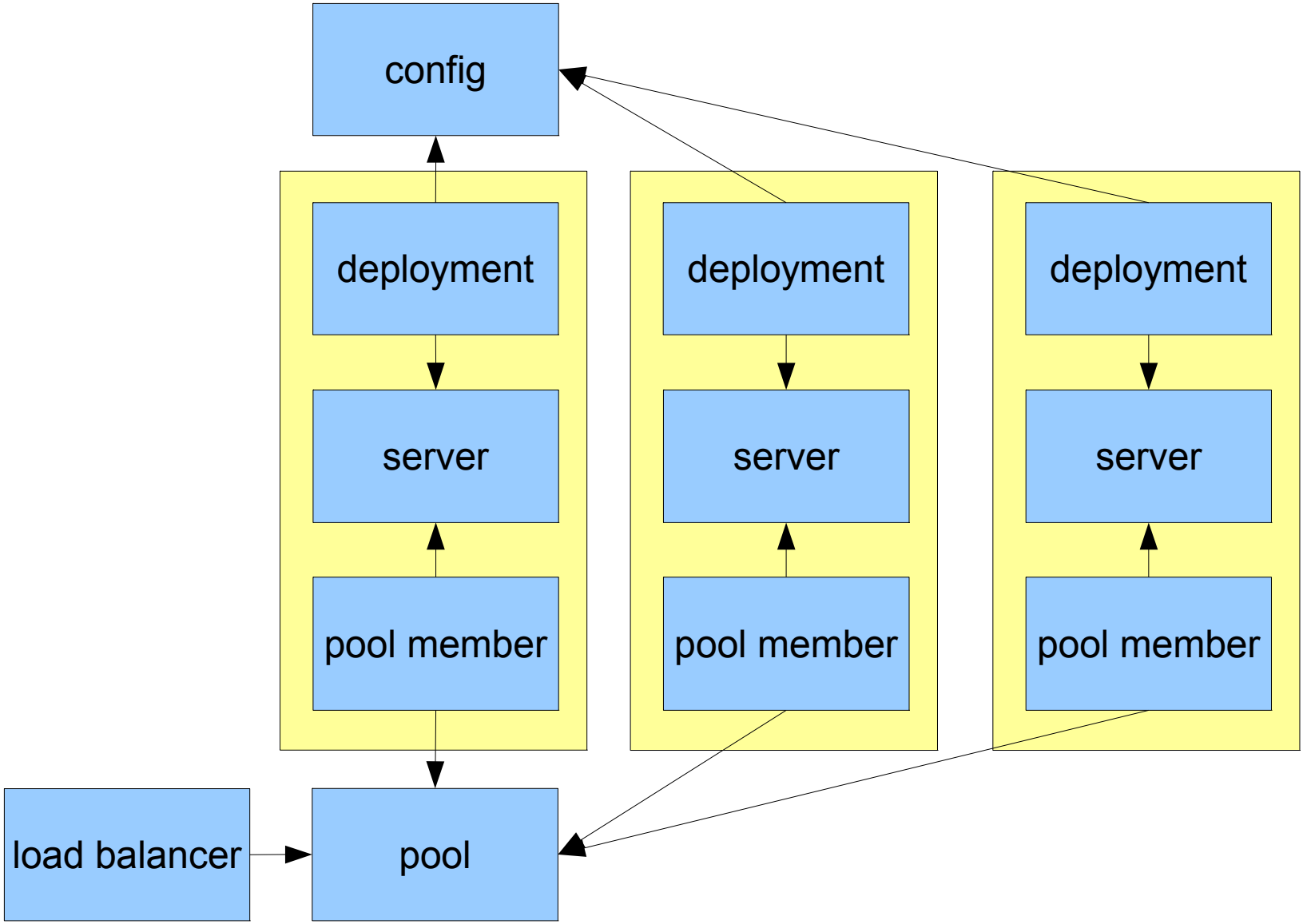
Deployments illustrated



Scaling deployments illustrated



Scaling deployments illustrated



Deployment extra inputs

- deploy_server_id
- deploy_action
- deploy_stack_id, deploy_resource_name
- deploy_signal_id
- deploy_auth_url, deploy_username, deploy_password, deploy_project_id, deploy_user_id

get_file intrinsic function

- python-heatclient fetches local files and URLs
- Contents of get_file calls included in heat stack-create request
- Initial support for including binary files

Script and cfn-init example

```
config:
  type: OS::Heat::StructuredConfig
  properties:
    group: cfn-init
    inputs:
    - name: bar
    config:
      config:
        files:
          /tmp/foo:
            content:
              get_input: bar
            mode: '000644'

check_tmp_foo:
  type: OS::Heat::SoftwareConfig
  properties:
    group: script
    outputs:
    - name: result
    config: {get_file: check_tmp_foo.sh}
```

```
deployment:
  type: OS::Heat::StructuredDeployment
  properties:
    name: 10_deployment
    signal_transport: NO_SIGNAL
    config:
      get_resource: config
    server:
      get_resource: server
    input_values:
      bar: baaaaa

deploy_check_tmp_foo:
  type: OS::Heat::SoftwareDeployment
  properties:
    name: 30_deploy_check_tmp_foo
    config:
      get_resource: check_tmp_foo
    server:
      get_resource: server
```

Script and cfn-init example

```
server:
  type: OS::Nova::Server
  properties:
    image: {get_param: image}
    flavor: {get_param: flavor}
    key_name: {get_param: key_name}
    security_groups:
    - {get_resource: the_sg}
    user_data_format: SOFTWARE_CONFIG
```

```
#!/bin/sh
echo -n "The file /tmp/foo contains `cat /tmp/foo` for server $deploy_server_id \
during $deploy_action" > $heat_outputs_path.result
```

Puppet example

```
config:
  type: OS::Heat::SoftwareConfig
  properties:
    group: puppet
    inputs:
    - name: foo
    - name: bar
    outputs:
    - name: result
    config:
      get_file: puppet-manifest.pp
```

```
deployment:
    type: OS::Heat::SoftwareDeployment
    properties:
      config:
        get_resource: config
      server:
        get_resource: server
      input_values:
        foo: fooooo
        bar: baaaaa
```

Puppet example

```
server:
 type: OS::Nova::Server
 properties:
   image: {get_param: image}
   flavor: {get_param: flavor}
   key_name: {get_param: key_name}
   security_groups:
   - {get_resource: the_sg}
   user_data_format: SOFTWARE_CONFIG
```

```
file {'barfile':
  ensure  => file,
  mode    => 0644,
  path    => "/tmp/$::bar",
  content => "$::foo",
}
file {'output_result':
  ensure  => file,
  path    => "$::heat_outputs_path.result",
  mode    => 0644,
  content => "The file /tmp/$::bar contains $::foo",
}
```

Image based example

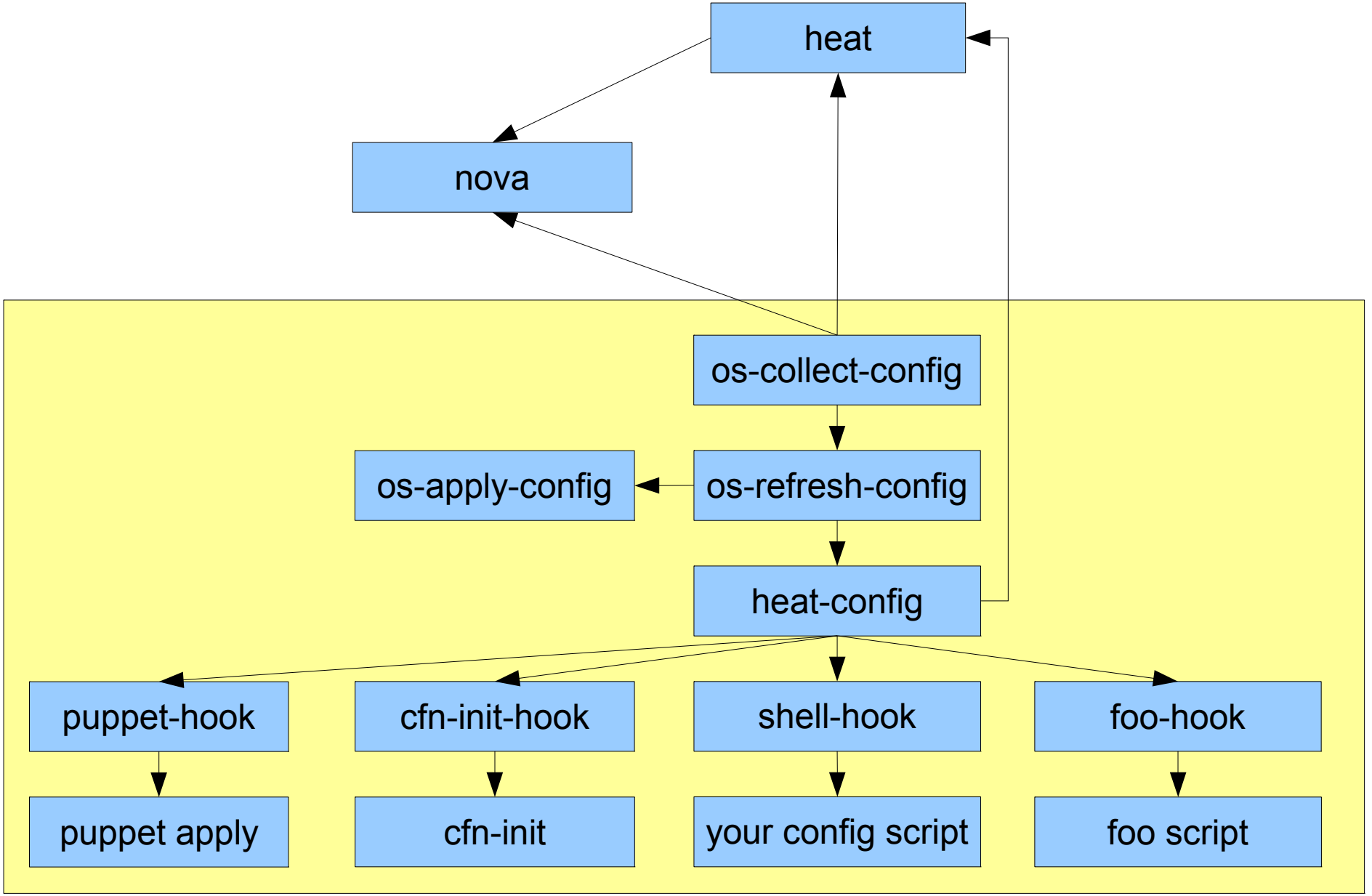
```
BlockStorageConfig:
  type: OS::Heat::StructuredConfig
  properties:
    group: os-apply-config
    config:
      cinder:
        db: {get_input: cinder_dsn}
        volume_size_mb: '5000'
        service-password:
          get_param: CinderPassword
        iscsi-helper:
          get_param: CinderISCSIHelper
      admin-password:
        get_param: AdminPassword
```

```
BlockStorage0Deployment:
  type: OS::Heat::StructuredDeployment
  properties:
    server: {get_resource: BlockStorage0}
    config: {get_resource: BlockStorageConfig}
    input_values:
      cinder_dsn:
        str_replace:
          template: |
            mysql://cinder:unset@address/cinder
          params:
            address:
              get_attr:
              - controller0
              - networks
              - ctlplane
              - 0
```

Hooks

- Consumes JSON from stdin, writes JSON to stdout
- Invokes configuration script with a particular tool
- Maps config inputs to some tool-specific concepts, e.g.
  - Environment variables (scripts)
  - Facts (puppet)
- Discovers output values after config tool execution

Hooks illustrated



Available hooks

- Script
- cfn-init
- Puppet
- Golden image configuration (not actually a hook!)

Hooks yet to write

- Chef
- Salt
- Ansible
- PowerShell

Mapping config model to tools

tool	config	inputs	outputs	options
script	the script	environment variables	files	-
cfn-init	declarative yaml	heat get_input	-	-
puppet	manifest	facts	files	extra modules?
chef solo	cookbook or recipe	attributes	attributes?	databags? cookbooks?
salt standalone minion	SLS	pillar data	grains? custom returner?	-
ansible connection local	playbook	variables	return data?	-
powershell	ps1 scripts	variables	Out-File?	-
image based	config yaml	heat get_input	os-refresh-config curl calls	-

Writing a hook

- Consumes JSON on stdin (inputs, script, options)
- Invokes the configuration tool to perform config
- Writes JSON to stdout (outputs)
- ~100 lines of python
- Contribute your hook to https://github.com/openstack/heat-templates

Golden image requirements

- os-collect-config
- os-refresh-config
- os-apply-config
- heat-config os-refresh-config scripts
- Hook for your chosen configuration tool
- Actual configuration tool

diskimage-builder for building disk images

```
  git clone https://git.openstack.org/openstack/diskimage-builder.git
  git clone https://git.openstack.org/openstack/tripleo-image-elements.git
  git clone https://git.openstack.org/openstack/heat-templates.git

  export ELEMENTS_PATH=\
tripleo-image-elements/elements:\
heat-templates/hot/software-config/elements

  diskimage-builder/bin/disk-image-create vm \
    fedora \
    heat-config \
    os-collect-config \
    os-refresh-config \
    os-apply-config \
    heat-config-script \
    heat-config-cfn-init \
    -o fedora-software-config.qcow2

  glance image-create --disk-format qcow2 --container-format bare \
    --name fedora-software-config < \
    fedora-software-config.qcow2
```

Whither the master configuration server?

- Heat *can* be the central source of truth, no master required
- No need for the complexity of syncing heat<->master, unless you really want to ;)
- ...or, minimal heat config could be used to hand off server to a config master

Planned improvements

- Other techniques for heat <-> server communication
  - Swift
  - Marconi
  - Servers in isolated tenant networks
- Action-aware config resource for alignment with TOSCA
- *Moar hoooks* (chef, salt, ansible, powershell...)
- Docker integration
- Windows support
- Deployments for shutdown in nova rebuild, reboot

Questions?

sbaker@redhat.com

https://wiki.openstack.org/wiki/Heat