### **Modify greeter release**

Though BOSH DNS provides a way of connecting jobs together without hardcoding instance IP addresses, we still have to rely on custom properties that contains instance dns names and are introduced only for purpose of connecting jobs. It would be much better if BOSH provides a unified mechanism for doing this kind of work. And with BOSH v2.0 such mechanism was introduced: it is called [BOSH links](https://bosh.io/docs/links.html)

When using links we need to explicitly specify what kind of service each job provides, and what kind of services it consumes. Four our greeter release we can specify that app job provides app link of type http and router job consumes this link. So let's start with updating our release.

1. Modify ~/greeter-release/jobs/app/spec to look like this

name: app  
templates:  
 ctl: bin/ctl  
  
packages:  
- greeter  
- ruby  
  
provides:  
- name: app  
 type: http  
 properties:  
 - port  
  
properties:  
 port:  
 description: "Port on which server is listening"  
 default: 8080

Pay attention to properties section of the link definition. With this section it is possible to add any property to the link. . We will later see how those properties are used by link consumer.

1. Modify ~/greeter-release/jobs/router/spec and add consumes section here

---  
name: router  
templates:  
 ctl: bin/ctl  
 config.yml.erb: config/config.yml  
  
packages:  
- greeter  
- ruby  
  
consumes:  
- name: app  
 type: http   
  
properties:  
 port:  
 description: "Port on which server is listening"  
 default: 8080

Pay attention that we also no longer need upstreams property because now our jobs are connected using links.

1. Modify ~/greeter-release/jobs/router/templates/config.yml.erb to use links

---  
<%=  
app\_link = link("app")  
port = app\_link.p("port")  
result = {}  
result["upstreams"] = app\_link.instances.map do |instance|  
 "#{instance.address}:#{port}"  
end  
JSON.dump(result)  
%>

Here we first obtain the link, then iterates over linked instances, create upstrems array and serialize it.

1. Finally we need to recreate the release and upload it to the director

cd ~/greeter-release  
bosh create-release --force  
bosh upload-release

### **Modify deployment manifest**

When using links not only release need to be updated. The following changes need to be made in the manifest.

1. upstrems property should be deleted, as router job no longer defines it
2. provides and consumes sections should be added to corresponding job definitions. In our case only one job provides links of type http so this step is optional. We can rely on [inplicit linking](https://bosh.io/docs/links.html#implicit). But we still want to make appropriate changes to the manifest just to show how it can be done.

~/deployment/greeter.v2.0.yml deployment manifest should look like the following

name: greeter-release  
  
releases:  
- name: greeter-release  
 version: latest  
  
instance\_groups:  
- name: app  
 instances: 1  
 azs: [z1]  
 vm\_type: t2.small  
 stemcell: ubuntu  
 jobs:  
 - name: app  
 provides:  
 app: {as: my\_app\_link}  
 properties: {}  
 networks:  
 - name: private  
  
- name: router  
 instances: 1  
 azs: [z1]  
 vm\_type: t2.small  
 stemcell: ubuntu  
 jobs:  
 - name: router  
 consumes:  
 app: {from: my\_app\_link}  
 properties: {}  
  
 networks:  
 - name: private  
 default: [dns, gateway]  
 - name: public  
 static\_ips:  
 - $REPLACE\_WITH\_ELASTIC\_IP  
  
stemcells:  
- alias: ubuntu  
 os: ubuntu-trusty  
 version: $REPLACE\_WITH\_STEMCELL\_VERSION  
update:  
 canaries: 1  
 canary\_watch\_time: 30000  
 update\_watch\_time: 30000  
 max\_in\_flight: 10  
 max\_errors: 1

And finally let's redeploy our release

bosh -n -d greeter-release deploy ~/deployment/greeter.v2.0.yml

When deployment is ready you can check if everything has been deployed as intended:

curl "http://$REPLACE\_WITH\_ELASTIC\_IP:8080"