


# Create the services

 All of the microservices builds will be identical to the ones in your **beta** environment. This is because both, beta and prod, are instances of the same service, and thus share the same repository id and pipeline (that listens for changes from the master branch). Alternatively, you could create separate services with different code repository branches, or create services using a service template that do not include a service pipeline

Let's provision the frontend service in our multi-svc production environment:

```
1 sh -c "~/environment/proton-workshop-helper add-instance --service_name frontend"
```

Please, provide the following values when prompted:

Input	Value
-------	-------

Template Name	lb-fargate-service
Environment Name	multi-svc-<your_account_id>
Port	3000
Desired Count	3
Task Size	medium
Scope	public
Image	<don't_change>
Environment Variables	CRYSTAL_URL=http://crystal.protonworkshop.prod.local:3000/crystal NODEJS_URL=http://nodejs.protonworkshop.prod.local:3000
Service Backends	crystal;nodejs

The proton-workshop-helper will a execute similar command to the following example:

```
1 aws proton update-service \  
2   --region ${AWS_REGION} \  
3   --name "frontend-prod" \  
4   --spec "proton: ServiceSpec  
5           instances:  
6             - name: 'frontend-beta'  
7               environment: 'multi-svc-beta'  
8               spec:  
9                 port: \"${Proton::CURRENT_VAL}\"  
10                desired_count: \"${Proton::CURRENT_VAL}\"  
11                task_size: \"${Proton::CURRENT_VAL}\"  
12                scope: \"${Proton::CURRENT_VAL}\"  
13                image: \"${Proton::CURRENT_VAL}\"  
14                env_vars: \"${Proton::CURRENT_VAL}\"  
15                service_backends: \"${Proton::CURRENT_VAL}\"
```



```
16     - name: 'frontend-prod'
17       environment: 'multi-svc-<your_account_id>'
18       spec:
19         port: 3000
20         desired_count: 3
21         task_size: medium
22         scope: public
23         image: public.ecr.aws/z9d2n7e1/nginx:1.19.5
24         env_vars: >
25           CRYSTAL_URL=http://crystal.protonworkshop.prod.local:3000/crystal;
26           NODEJS_URL=http://nodejs.protonworkshop.prod.local:3000" \
```

It is time to repeat the same process with crystal and nodejs.

Crystal

Nodejs

Input	Value
Service Instance Name	crystal-prod
Template Name	1h-frontend-service

[Report Workshop Studio bug](#)

Environment Name	multi-svc-<your_account_id>
Port	3000
Desired Count	3
Task Size	medium
Scope	private
Image	<don't_change>
Environment Variables	<leave_empty>
Service Backends	<leave_empty>

Execute the proton-workshop-helper:

```
1 sh -c "~/environment/proton-workshop-helper add-instance --service_name crystal"
```