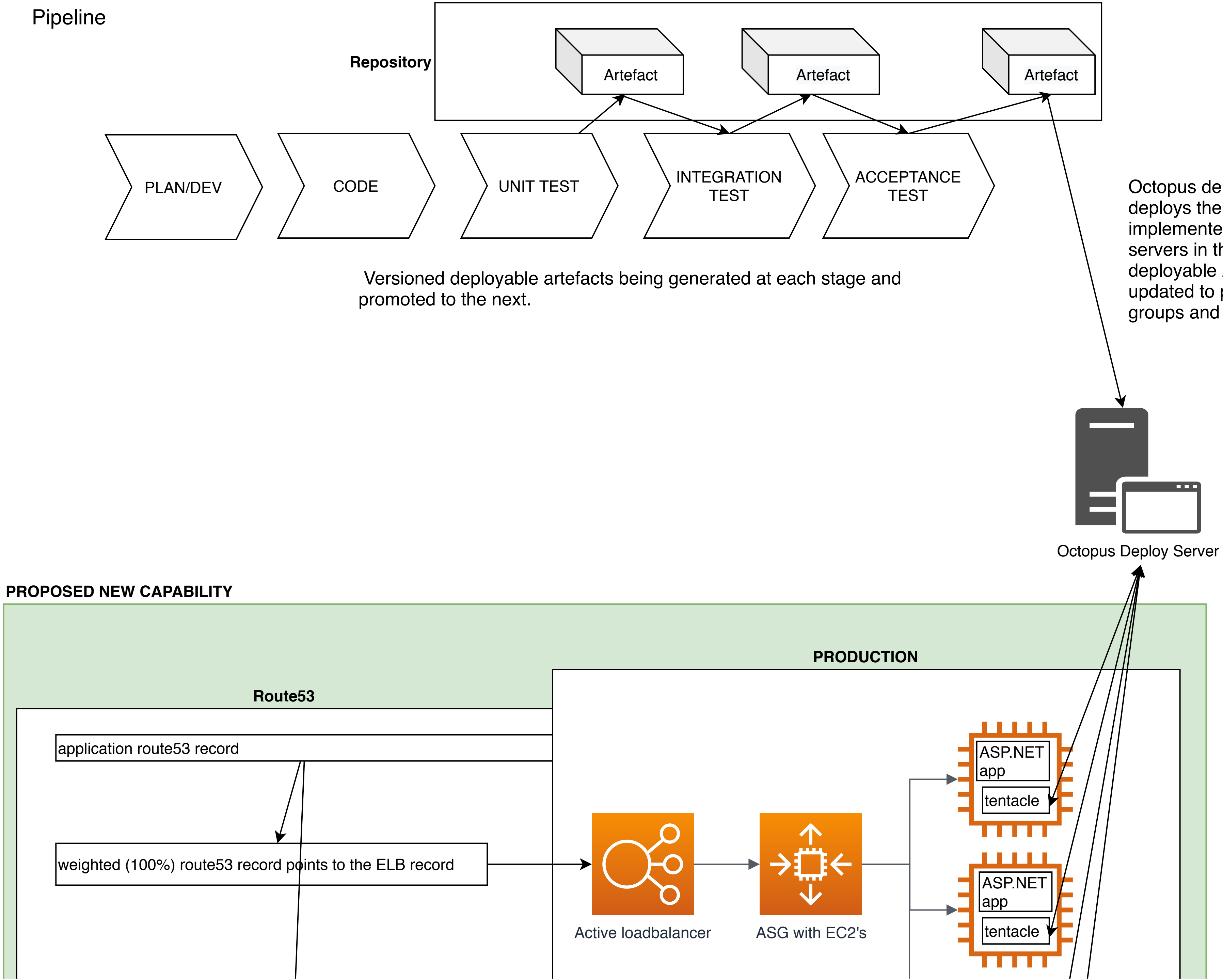
Assumption - Current capability

Pipeline



Versioned deployable artefacts being generated at each stage and promoted to the next.

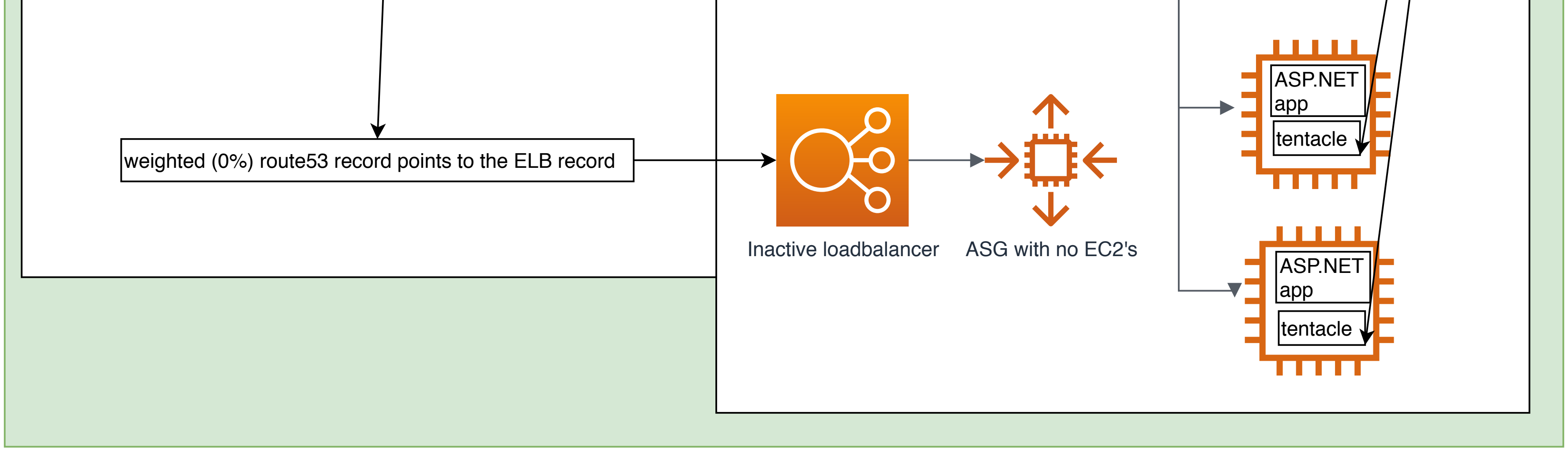
Octopus deployment server pulls the latest arte
deploys them to the EC2's. Any method can be
implemented from downloading and loading up
servers in the runtime to pre-baking/preparing
deployable AIM's. ASG's/Launch configurations
updated to prepare for the process or create ne
groups and attaching them.

Octopus deployment server com
to the tentacles and deploys the
when triggered

efact and
e
o the

s can be
ew target

nmunicates
changes



Route53 holds the weighted records for both the load balancers. Load balancers here are to provide reverse proxy. Once the deployment happens the ASG for the old version can be reduced in the count for EC2's and those EC2's are untouched.

The inactive ASG can be made active with EC2's having latest artefacts deployed, route53 weight for the second load balancer can be slowly increased as the ASG's bring in new healthy nodes.
ELB health checks or route53 health checks can be utilized to determine the health of the new nodes, and making the weight for the second record to a 100 percent which completes the deployment.

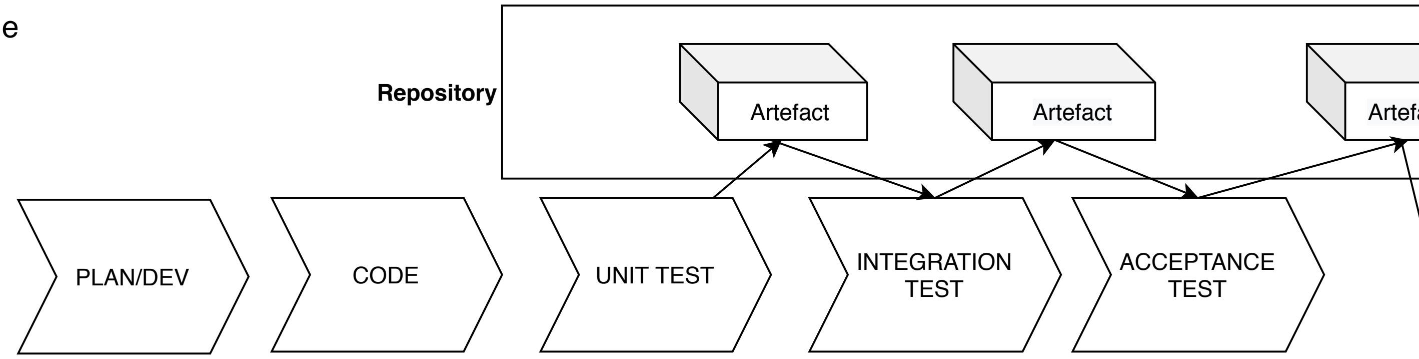
In the target state the current active ELB/ASG will become inactive and vice versa.

The method seems to provide a graceful change with no downtimes.

P

Assumption - Current capability

Pipeline



Repository

Artefact

Artefact

Artef

PLAN/DEV

CODE

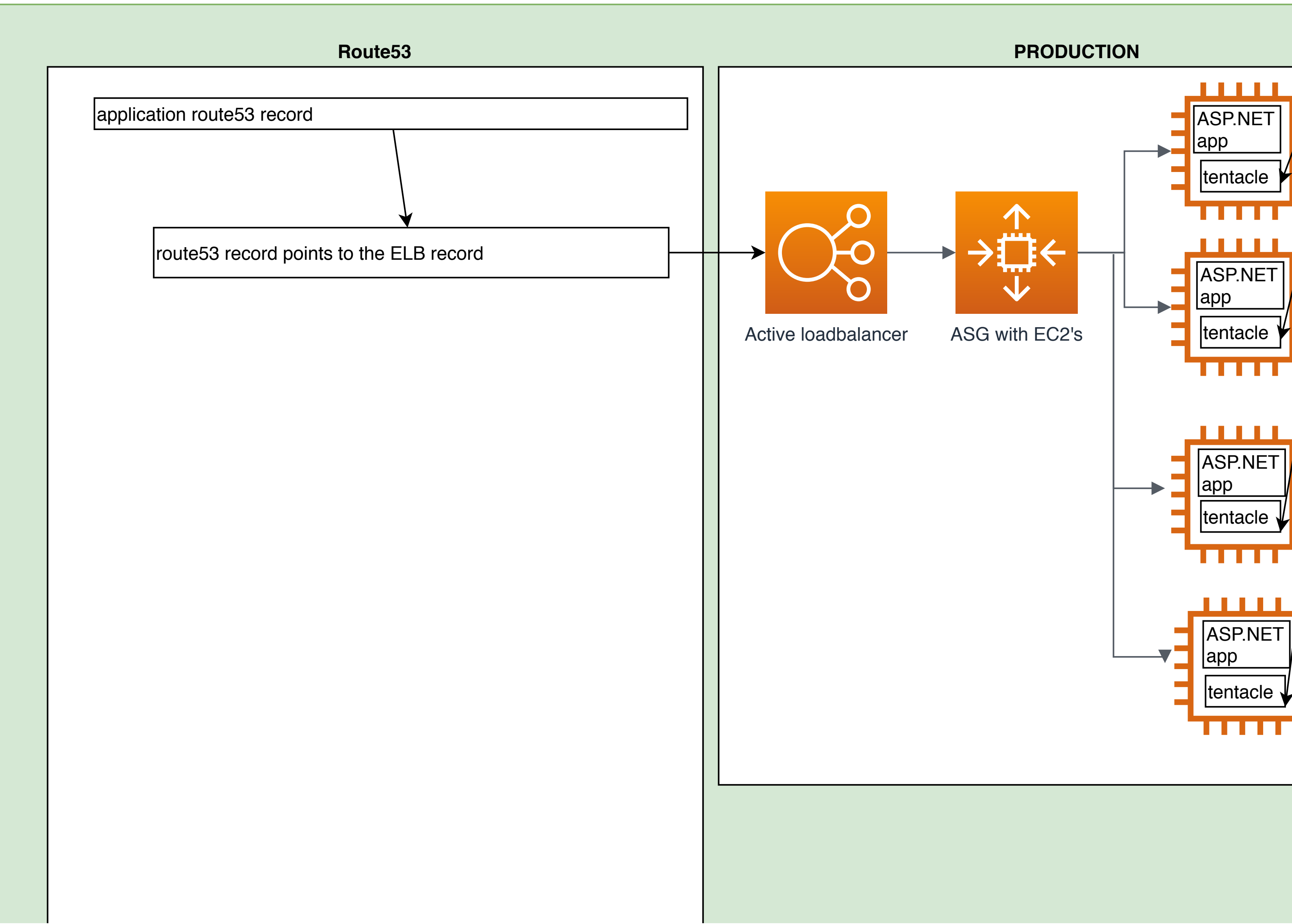
UNIT TEST

INTEGRATION TEST

ACCEPTANCE TEST

Versioned deployable artefacts being generated at each stage and promoted to the next.

ROPOSED NEW CAPABILITY



Route53

application route53 record

route53 record points to the ELB record

PRODUCTION

Active loadbalancer

ASG with EC2's

ASP.NET app

tentacle

ASP.NET app

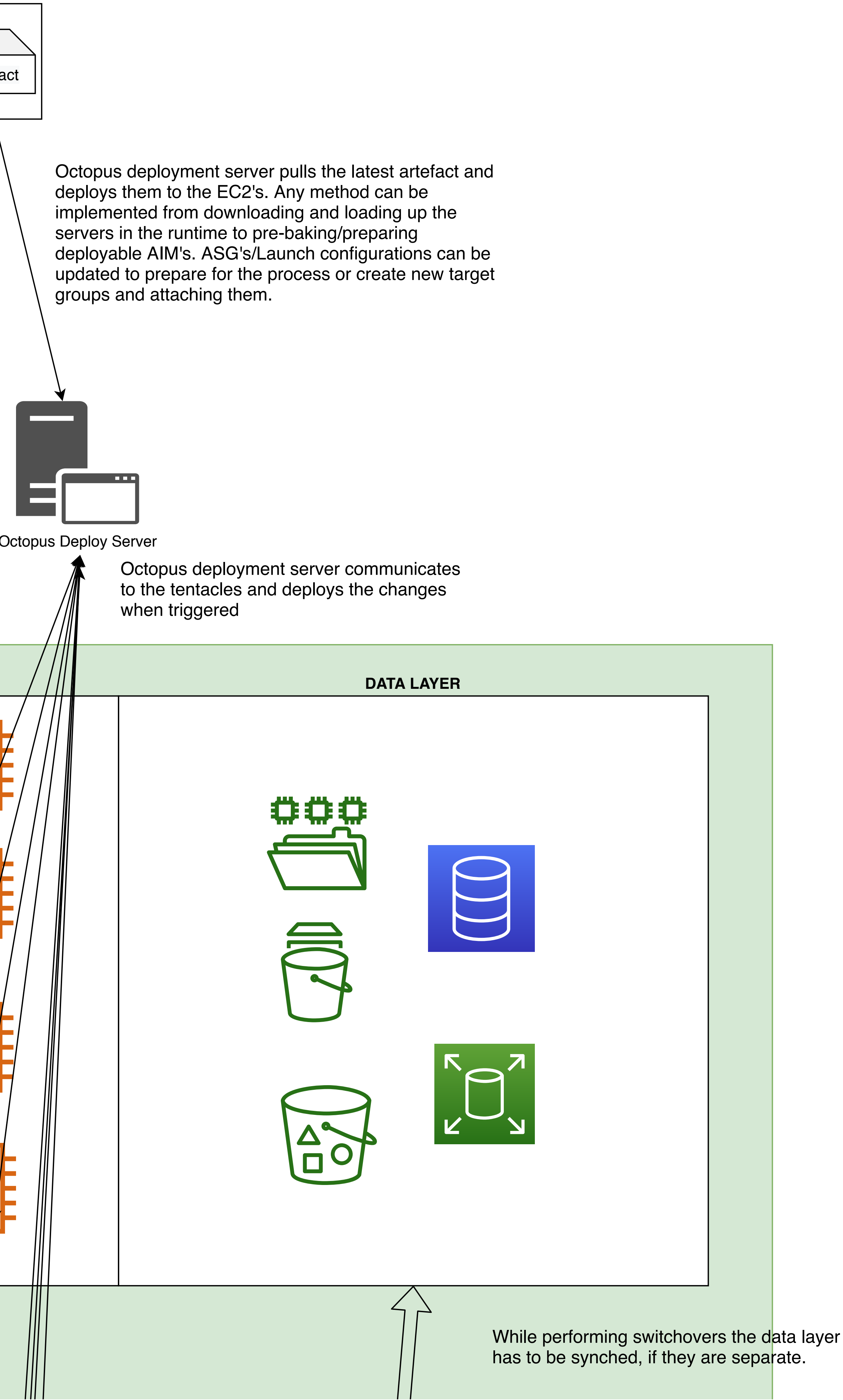
tentacle

ASP.NET app

tentacle

ASP.NET app

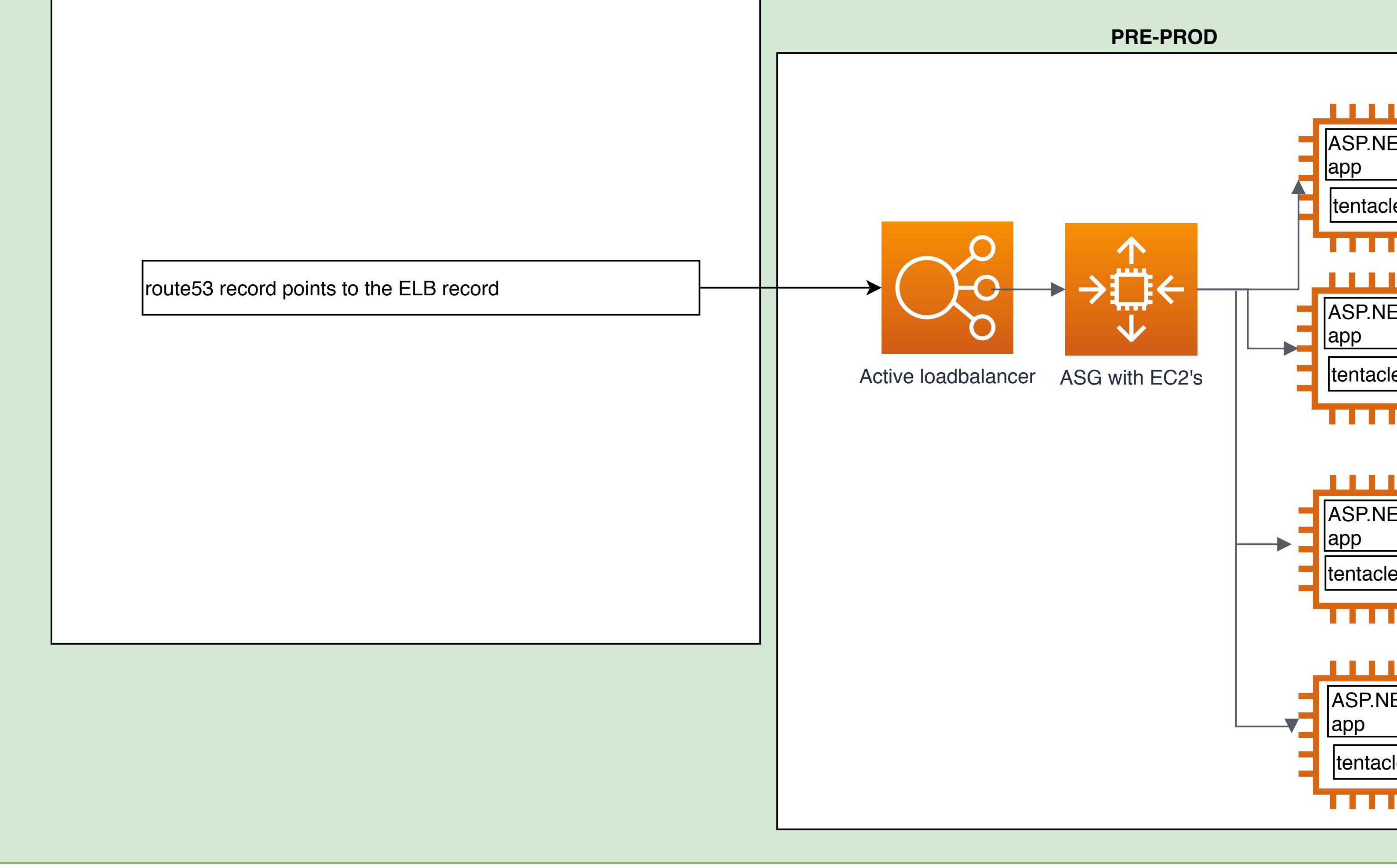
tentacle



Octopus deployment server pulls the latest artefact and deploys them to the EC2's. Any method can be implemented from downloading and loading up the servers in the runtime to pre-baking/preparing deployable AIM's. ASG's/Launch configurations can be updated to prepare for the process or create new target groups and attaching them.

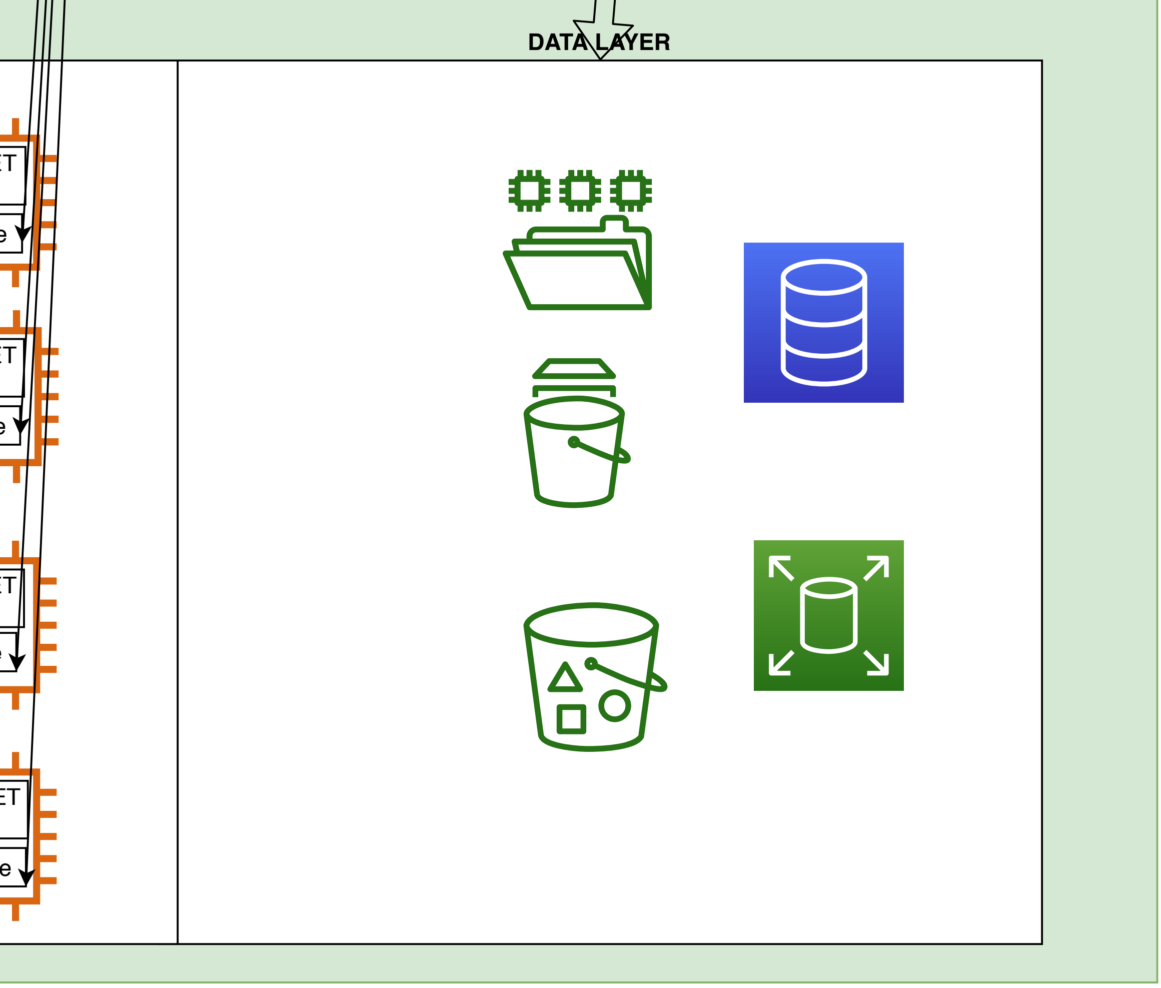
Octopus deployment server communicates to the tentacles and deploys the changes when triggered

While performing switchovers the data layer has to be synched, if they are separate.



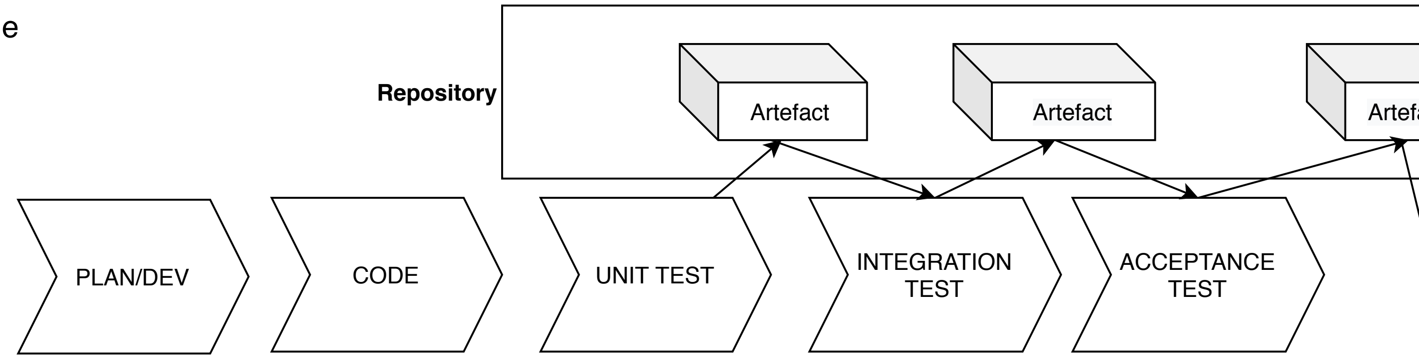
Here in this solution its a classic blue green deployment, when the pre-prod environment is all ready to switch, with latest code and all the data is synced up if they are not shared change the route 53 to point to the pre-prod.

In the target state the current pre-prod becomes prod and vice versa. All the code deployments will be pointed towards the pre-prod and Continuously deploy to the pre-prod and when ready sync the data part up and switch again.


DATA LAYER
ET
e
ET
e
ET
e
ET
e

Assumption - Current capability

Pipeline



Repository

Artefact

Artefact

Artef

PLAN/DEV

CODE

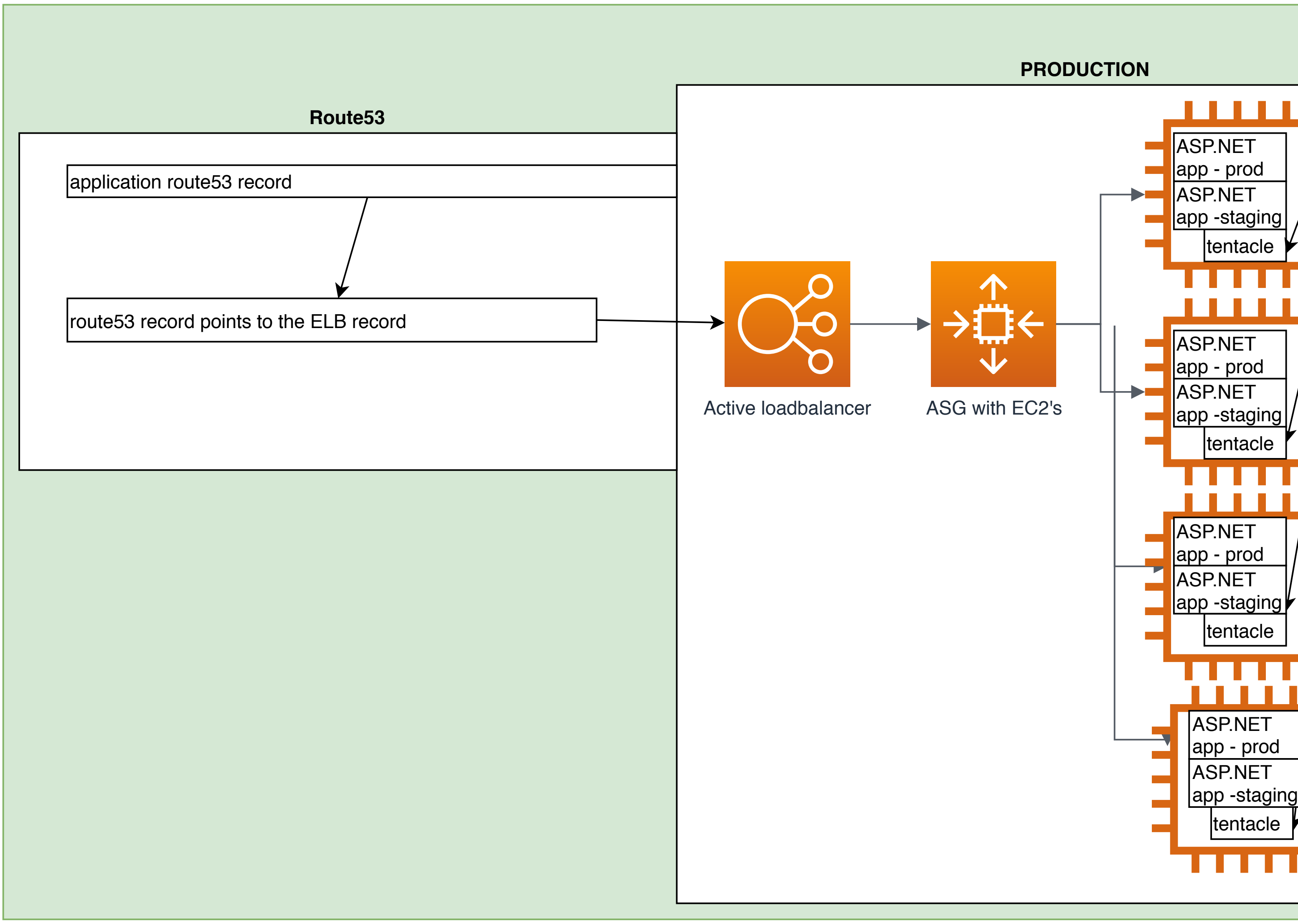
UNIT TEST

INTEGRATION TEST

ACCEPTANCE TEST

Versioned deployable artefacts being generated at each stage and promoted to the next.

PROPOSED NEW CAPABILITY



PRODUCTION

Route53

application route53 record

route53 record points to the ELB record

Active loadbalancer

ASG with EC2's

ASP.NET app - prod

ASP.NET app -staging

tentacle

ASP.NET app - prod

ASP.NET app -staging

tentacle

ASP.NET app - prod

ASP.NET app -staging

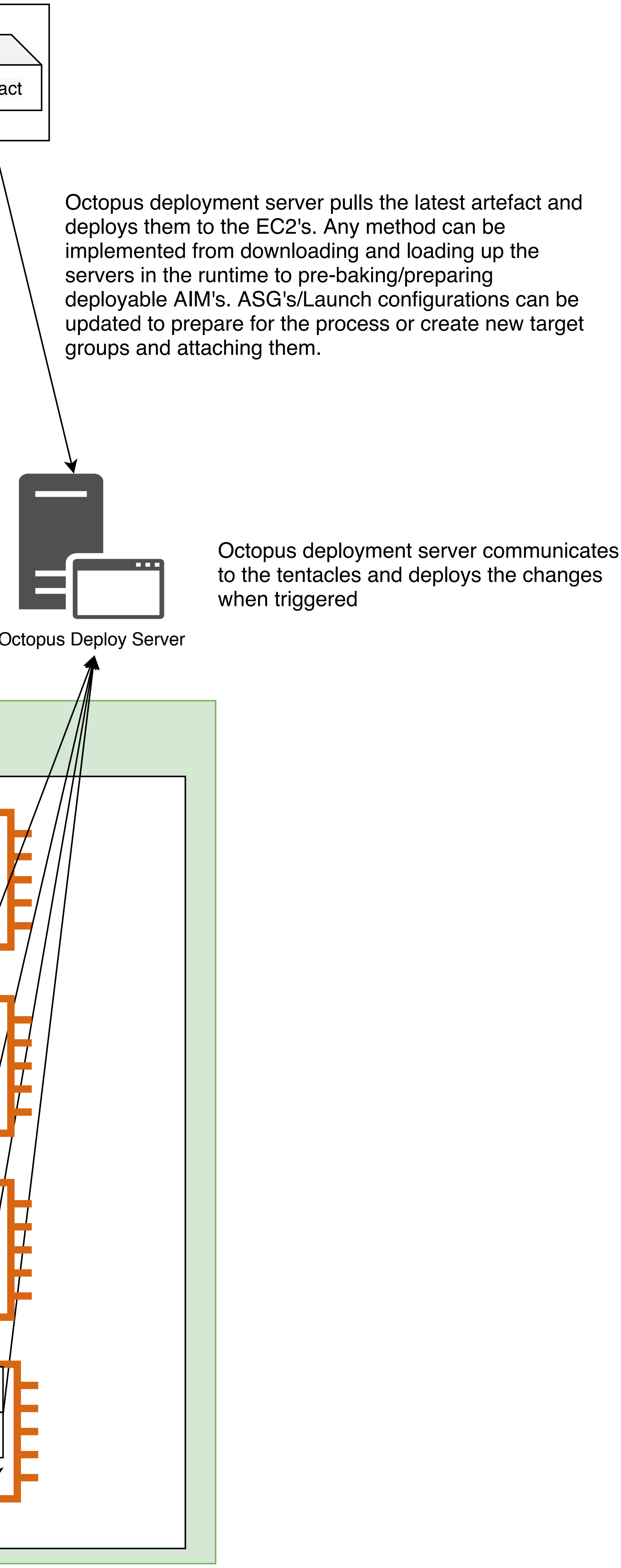
tentacle

ASP.NET app - prod

ASP.NET app -staging

tentacle

In this solution we are not using much of the infra components but rather another copy of the application itself within the same EC2 host.



Octopus deployment server pulls the latest artefact and deploys them to the EC2's. Any method can be implemented from downloading and loading up the servers in the runtime to pre-baking/preparing deployable AIM's. ASG's/Launch configurations can be updated to prepare for the process or create new target groups and attaching them.

Octopus deployment server communicates to the tentacles and deploys the changes when triggered

When ready to switch the Octopus deploy can execute script on the remote server. The actual implementation could be anything ranging for switching the files/folder where the application files are stored to having another copy of the webserver running/switching when triggered.

Though this method uses less AWS infra components there could be some draw backs and hence included in the last. Notably the below ones.

- The shared resources in the servers if any such as ports or pipes, they could conflict if they are not released in time.

-This may confuse the other integrations the application may have if they come up different identities / signatures.

-Any licensable components /dependencies may experience issues.

-Chances are there overtime some manual fixes/slack may come into existence.