Question 1

Please see `deployment.yaml` and `service.yaml`

Following is the output of some of the checker commands for your convenience

- `kubectl get all`

```
NAME                                        READY   STATUS    RESTARTS   AGE
pod/rabbitmq-deployment-7ff8c859c6-2qwhr    1/1     Running   0          69m
pod/rabbitmq-deployment-7ff8c859c6-p94bl    1/1     Running   0          69m
pod/rabbitmq-deployment-7ff8c859c6-vh7d2    1/1     Running   0          69m
pod/rabbitmq-deployment-7ff8c859c6-zgqxg    1/1     Running   0          69m
pod/rabbitmq-deployment-7ff8c859c6-zxh8r    1/1     Running   0          69m

NAME                     TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
          AGE
service/kubernetes       ClusterIP      10.152.183.1     <none>        443/TCP
          32h
service/rabbitmq-svc     LoadBalancer   10.152.183.208   <pending>
32500:32500/TCP   70m

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/rabbitmq-deployment     5/5     5            5           69m

NAME                                               DESIRED   CURRENT   READY
AGE
replicaset.apps/rabbitmq-deployment-7ff8c859c6     5         5         5
69m
```

- `kubectl describe deployment.apps/rabbitmq-deployment`

```
Name:                   rabbitmq-deployment
Namespace:              default
CreationTimestamp:      Wed, 16 Oct 2019 18:25:56 -0500
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
                        kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":
{},"name":"rabbitmq-deployment","namespace":"default"},"spec":{"repl...
Selector:               app=rabbit
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0
unavailable
```

```
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=rabbit
  Containers:
   rabbitmq:
    Image:      rabbitmq:3-management
    Port:       15672/TCP
    Host Port:  0/TCP
    Limits:
      cpu:     125m
      memory:  300Mi
    Requests:
      cpu:        50m
      memory:     150Mi
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   rabbitmq-deployment-7ff8c859c6 (5/5 replicas created)
Events:          <none>
```

- `kubctl describe service/rabbitmq-svc`

```
Name:                     rabbitmq-svc
Namespace:                default
Labels:                   app=rabbit
Annotations:              kubectl.kubernetes.io/last-applied-configuration:
                            {"apiVersion":"v1","kind":"Service","metadata":
{"annotations":{},"labels":{"app":"rabbit"},"name":"rabbitmq-
svc","namespace":"default"},"s...
Selector:                 app=nginx
Type:                     LoadBalancer
IP:                       10.152.183.208
Port:                     <unset>  32500/TCP
TargetPort:               15672/TCP
NodePort:                 <unset>  32500/TCP
Endpoints:                <none>
```

```
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Question 2

For the given `interesting.pcap` file, I noticed that there are a huge amount of [SYN] signals sent to the same target from different IP addresses. My assumption is someone is conducting a DOS attack to the server, which aims to block the three way handshake from other clients.

Question 3

```
policy_module(bank, 1.0.0)
type bank_exec_t;
type bank_admin_t;
type bank_balance_t;
init_daemon_domain(bank_admin_t, bank_exec_t)
files_type(bank_balance_t)
require {
    type bank_admin_t;
    type bank_balance_t;
    type user_home_t;
    class file { create execute open read write getattr };
    class dir { add_name search write create remove_name getattr };
}
type_transition initrc_t bank_exec_t:file bank_exec_t;
allow user_home_t bank_balance_t:dir { read getattr };
allow bank_admin_t bank_balance_t:file { read write delete create getattr };
allow bank_admin_t bank_balance_t:dir { create getattr };
allow user_home_t user_home_t:dir { create open read write getattr };
```

Question 4

- First I used this command to find the syscalls that is related to /var/www

  `sudo sysdig -r suspicious.scap | grep /var/www`

- Then I located the PIDs of the process that makes calls to /var/www, which is `18886, 18888, 18864, 18865, 18866,`. Then I used the following command to see the exact syscalls that were made by that process:

- In order to make sure the above processes are actaully making large network bandwidth usage, I ran the following command: `sysdig -r suspicious.scap -c topprocs_net`, which gave me the following result: (* are the pids that we found suspicious)

```
Bytes               Process             PID
```

```
--------------------------------------------------------------------------------
---
634.46KB            kube-apiserver      6314
240.79KB            etcd                6007
176.45KB            heapster            9798
139.40KB            influxd             9965
131.46KB            kube-controller     6605
64.88KB             kubelet             6349
46.69KB             coredns             8870
37.94KB             kube-scheduler      6603
35.25KB             pod_nanny           10647
35.25KB             pod_nanny           10442
20.54KB             kube-proxy          6320
18.77KB             Socket              7179
13.17KB             systemd-resolve     452
12.17KB             dashboard           10031
9.07KB              apache2             18866 *
3.85KB              apache2             18864 *
3.85KB              apache2             18888 *
3.14KB              apache2             18886 *
2.98KB              DNS                 7179
1.47KB              mysqld              19358
931B                apache2             18859
804B                apache2             18865 *
395B                eventer             10193
235B                NetworkManager      636
```

- Finally, in order to findout where the traffic is comming from, I ran the following command to check the ip addresses of each processes (I used system call accept() as filter because it specifies the source IP and target IP as its inputs:

  ```
  sudo sysdig -r suspicious.scap \
  proc.pid=18886 or \
  proc.pid=18888 or \
  proc.pid=18864 or \
  proc.pid=18865 or \
  proc.pid=18866 \
  | grep accept
  ```

  The above scripts generates the following result:

```
174425 17:51:08.958745443 0 apache2 (18886) < accept
fd=10(<4t>10.1.1.1:43692->10.1.1.28:80) tuple=10.1.1.1:43692->10.1.1.28:80
queuepct=0 queuelen=0 queuemax=128
434940 17:51:13.971452480 0 apache2 (18886) > accept flags=0
460020 17:51:14.726287385 1 apache2 (18888) < accept
fd=10(<4t>10.1.1.1:43712->10.1.1.28:80) tuple=10.1.1.1:43712->10.1.1.28:80
queuepct=0 queuelen=0 queuemax=128
663023 17:51:19.734840292 1 apache2 (18888) > accept flags=0
987491 17:51:28.024501242 0 apache2 (18864) < accept
fd=10(<4t>10.1.1.1:43762->10.1.1.29:80) tuple=10.1.1.1:43762->10.1.1.29:80
queuepct=0 queuelen=0 queuemax=128
1198435 17:51:33.032192827 0 apache2 (18864) > accept flags=0
1229246 17:51:33.537011565 0 apache2 (18865) < accept
fd=10(<4t>10.1.1.1:43780->10.1.1.29:80) tuple=10.1.1.1:43780->10.1.1.29:80
queuepct=0 queuelen=0 queuemax=128
1424070 17:51:38.543233202 0 apache2 (18865) > accept flags=0
1522863 17:51:40.930099309 0 apache2 (18866) < accept
fd=10(<4t>10.1.1.1:43804->10.1.1.29:80) tuple=10.1.1.1:43804->10.1.1.29:80
queuepct=0 queuelen=0 queuemax=128
```

And we can see that the traffic all comes from this same IP address: `10.1.1.1`

In order to get more information, I ran sysdig-inspect and used the following filter command:

`proc.pid=18886 or proc.pid=18888 or proc.pid=18864 or proc.pid=18865 or proc.pid=18866`

PID	VPID	CPU	USER	TH	VIRT	RES	FILE	NET	CONTAINER	Command
18888	29	0	www-data	1	85848064	15392768	2253	3949		/usr/sbin/apache2 -k start
18866	30	0	www-data	1	85975040	15884288	12596	9290		/usr/sbin/apache2 -k start
18886	27	0	www-data	1	85966848	15511552	2159	3223		/usr/sbin/apache2 -k start
18864	28	0	www-data	1	85966848	15810560	2253	3949		/usr/sbin/apache2 -k start
18865	29	0	www-data	1	85975040	15683584	675	806		/usr/sbin/apache2 -k start