Salam dostlar, yəqin ki hər biriniz bu texnologiya haqqında eşitmisiniz, lakin əminəm ki, yaxşı şəkildə izah edən nə bir video nədəki bir məqalə tapmısız. Bu məqalədə məhz mən sizə Docker-ın həqiqətdə nə olduğunu bir-bir izah edəcəyəm. Elə isə başlıyaq.

Başlıyacağımız ilk hissə əlbətdəki “Docker nədir?” sualına cavab vermək olacaqdır.Docker app-larımızı izolasiya sistemində çalışdıra bilən bir texnologiyadır. Sadə deyimlə desək Docker bizə, yəni biz developerlərə yaratdığımız application-ı production enviroment-də hər bir kəsin(bunlara misal tester, digər developerlər aiddir) problemsiz şəkildə test edə bilməyinə şərait yaradır. Diqqət yetirdinizsə burada problemsiz şəkildə sözündən bəhs etdim. Məgər bizim yaratmış olduğumuz application production enviroment-ə çıxdıqda problemlərə gətirib çıxardır onu test edərkən. Cavab bəli! Əgər elədirsə o hansı problemlədir ki, onu bilmək olarmı sualını əminəm hər birinizi maraqlandırır. Elə isə gəlin o problemlər nələrdir bir-bir baxaq.

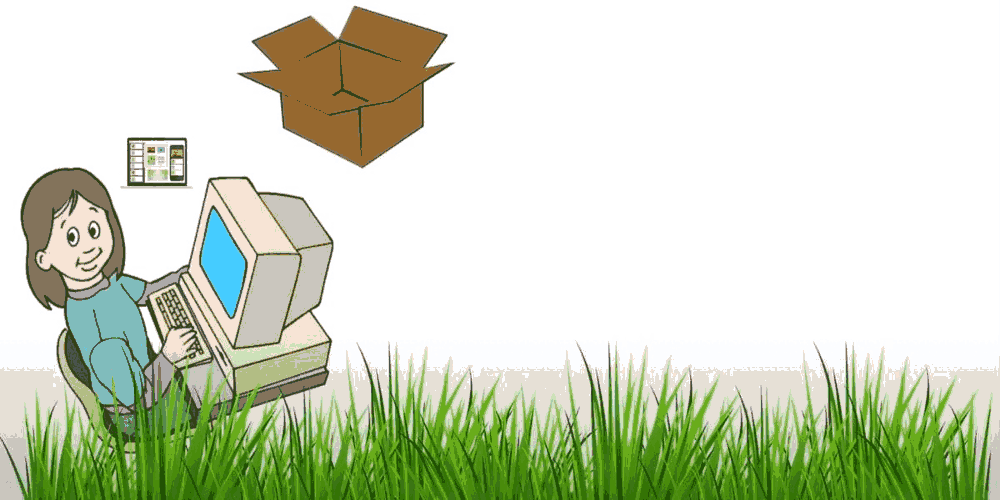
Fərz edin bizim java əsaslı web application-mız vardır və bunu production enviroment-ə çıxardmışıq ki, onu məsələn işlədiyiniz mühitdə hər hansı bir tester ki, adətən application-ı testerlər test edir, adından da məlum olduğu kimi və yaxud digər developerlərin test edə bilsin deyə. Fərz edin bizim web-application özü ilə bərabər əlavə texnologiyalar istifadə edir: Məsələn: Database tool olaraq MySQL, cache əməliyyatı üçün REDİS, loglama əməliyyatı üçün KİBANA və s. bunları saysaq bitməz kimi texnologiyalar bizim web-application da olmazsa olmazdır. Və burada aşağıdakı problemlərə gətirib çıxardacaqdır.

1. Yuxarıda sadalanan hər hansı bir tools web-application-mızı istifadə edəcək client-ın kompüterində olmaya bilər və yaxud olsa belə versiyaları toqquşa bilər.
2. Daha sonra istifadə etdiyimiz web-application da saysız hesabsız dependencie-lər olur ki, yenədə bu client-ın kompüterində olmaya bilər və yaxud olsa belə versiyaları toqquşa bilər.
3. Ən sonunda isə istifadə etdiyimiz web-application da saysız hesabsız library-lər ola bilər, hansıki yenədə bu client-ın kompüterində olmaya bilər və yaxud olsa belə versiyaları toqquşa bilər.

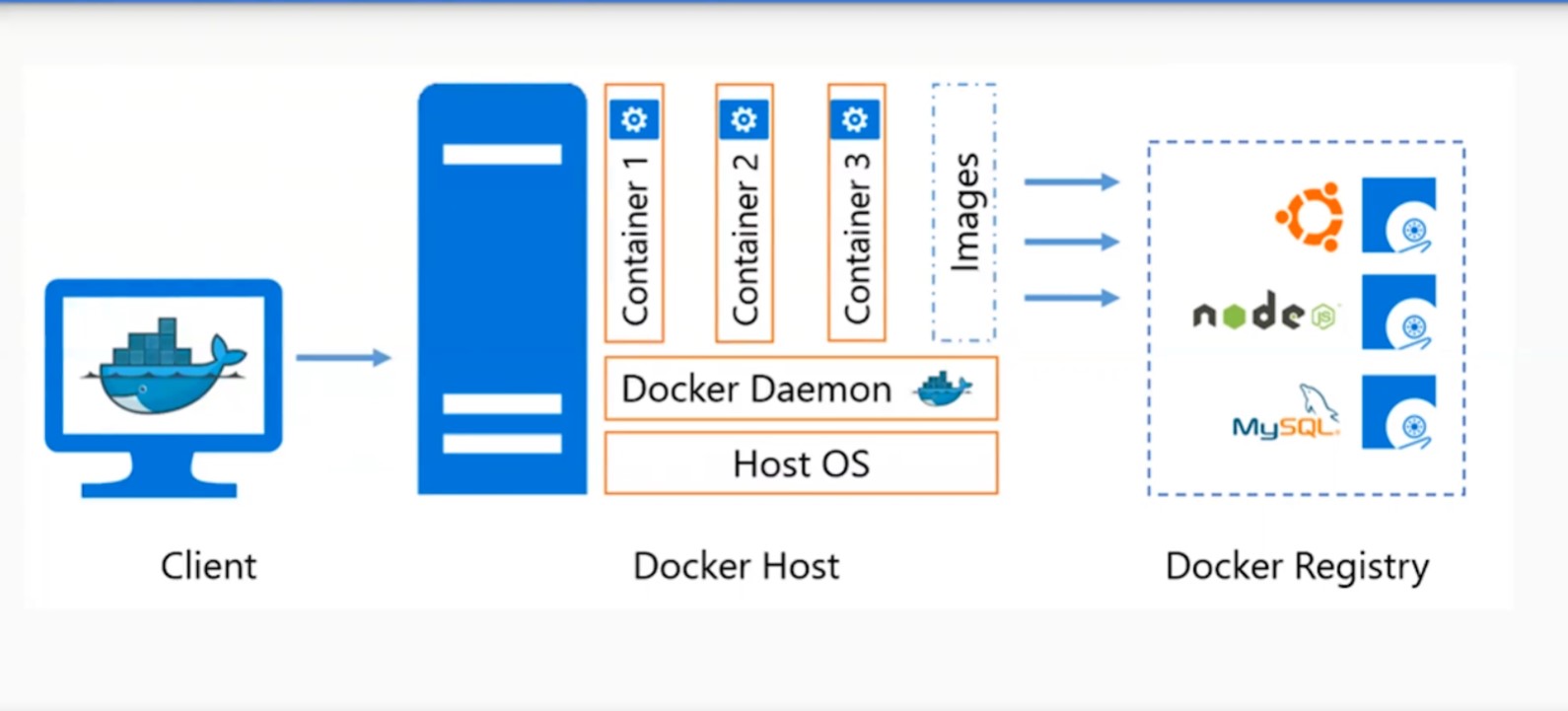
Hazır olun əsl surpriz buradadır. Baxın yuxarıda sadlanan bu problemlərin həll olunmağından ötəri Docker bizə mükəmməl şəkildə şərait yaradır. Necə? Gəlin buna biryerdə baxaq.

Docker deyir ki, sən developer mən bilirəm ki, sən bir web-application yaratmısan və onun içində ona lazım olacaq tools-lar, dependencie-lər və library-lər vardır ki, bunlar mütləq şəkildə bir yerdə başın qoyacaqdır sənin web-application-nını test edən client-ın kompüterində. Ona görə sən gəl mənim container-imdən istifadə edərək bu problemə birdəfəmlik şəkildə kökündən son qoyaq. Necə? Elə isə gəlin biryerdə baxaq.

Docker bizə container texnologiyasını verir ki, adından da məlum olduğu kimi container, yəni bir qablaşdırma və yaxud qutu kimi başa düşə bilərsiniz. Bu container vasitəsi ilə biz onun içində bizim production enviroment də test olunacaq web-application-mız da olan tools-ları hansıki yuxarıda sadalamışam, onları bu container-in içinə atırıq və sonrasında biz həmən container-i hub docker-a yaxud docker registry-ə əlavə etməyimiz bəs edir. Gördünüzmü necədə sadə? Sadəcə bir container vasitəsi ilə biz bu problemi aradan qaldırmış olduq. Ən sonunda isə sənin hub docker-a yükləmiş olduğun container-i istənilən tester yaxud developer ki, əlbətdə onlarında az-çox docker texnologiyasından başı çıxmalıdır, onlar nə edirlər sadəcə sənin atmış olduğun containeri-i pull edərək ən sonrasında run edib çalışdırırlar və beləliklə sənin web-application-ın istənilən client-ın kompüterində problemsiz şəkildə çalışır. Belə dostlar bu bizdə oldu “Docker nədir?” sualına cavab. Aşağıdakı şəkildən rahat şəkildə nə yazdığımı anlaya bilərsiniz.



1. Docker bir conteynirayzd texnologiyadır. Conteynirayzd o deməkdir ki, siz app-larıvızı qutular şəklinə salaraq onları biri-birindən ayırı şəkildə yəni isolation şəklində çalışdırırsınız deməkdir. Bu nə deməkdir, bu o deməkdir ki, siz app-larınızı docker vasitəsi ilə conteynır halına salaraq, istənilən mühit də istənilən clinet-lar tərəfindən problemsiz şəkildə app-ınızın test edilə bilməsini təmin edirsiniz. Bunu Docker da, yəni sizin app-larınızın problemsiz şəkildə istənilən əməliyyat sistemində və yaxud istənilən clientın, kimki sizin app-nızı test edəcək onların kompüterində problemsiz şəkildə çalışa bilməsini təmin edən məhz Docker-ın bizə sağlamış olduğu container-dır. Docker-ı bir real həyatdan yekə gəmi kimi başa düşə bilərsiniz, real həyatda okeanlarda səyahət edən yekə gəmi nə edir, yəqinki görmüsən özündə yekə konteynırlar daşıyır və sonrasında yekə gəmi limanlara çatdıqda bu konteynırlar həmən limanlara daşınaraq ona lazım olan ölkələr tərəfindən problemsiz şəkildə istifadə olunur, bax həmən Docker texnologiyasını da okenalar da yer alan yekə gəmi kimi başa düşə bilərsiniz ki, həmən onun daşıdığı konteynırların içində sizin app-larınız vardır və o applar-ınız yekə gəmi limana çatdıqda ona lazım olan ölkələr tərəfindən problemsiz şəkildə istifadə olunmasını sizin app-nızı istifadə edən clinet-lar kimi düşünə bilərsiniz. Qısası Docker bizə deyir ki, Build once, Run anywhere. Yəni birdəfə proyekt yarat və o proyekti həryerdə run elətdirə bil. Günümüz də Conteynirayzd texnologiyaları çoxdur, lakin Docker bunların ən məhşurudur.

**Docker Components**

Gördüyümüz kimi yuxarıdakı şəkil 3 hissədən ibarətdir:

**Client**

Bu hissəni bizim terminalda yazmış olduğumuz komandaların icra olunduğu kimi başa düşə bilərik. Biz bu komandalar vasitəsi ilə nə edirik ortada duran şəkildə olan hissə ilə əlaqəyə girərərk, Docker Host da Docker Daemon-ı hərəkətə gətirərək, bizim Client hissəsində yazmış olduğumuz komandalar ilə Docker Daemon ilə əlaqəyə girərək müxtəlif prosesləri icra etmiş oluruq.

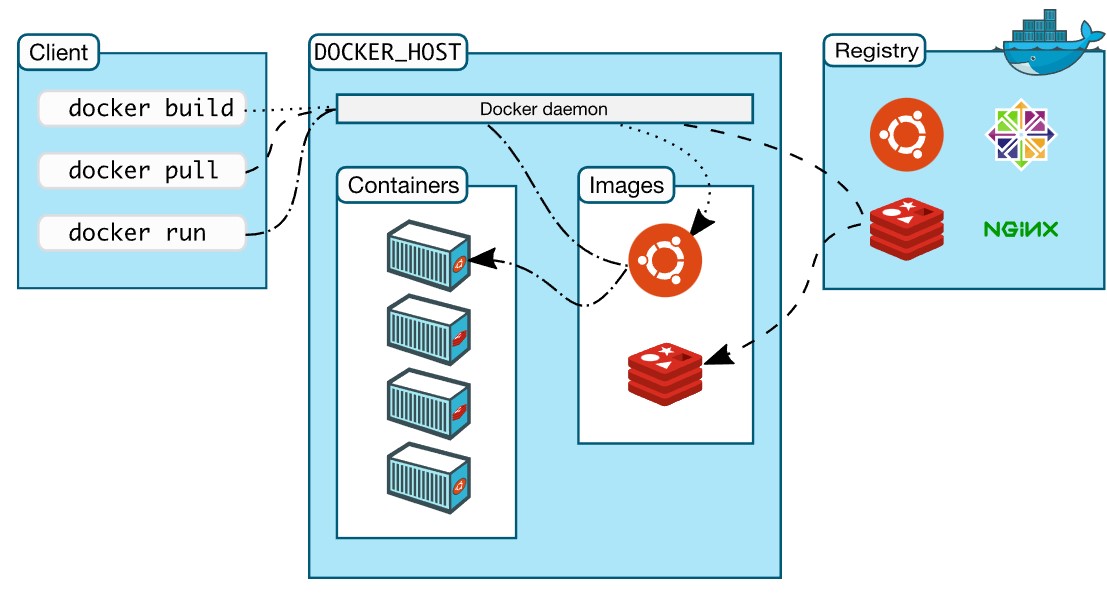
**Docker Host**

Bu hissəni belə başa düşə bilərik ki, biz Docker-ı kompüterimizə insall etdiyimiz an kompüterimizdə yer alan və Client-dan gələn komandaları icra edə bilən və həmçinin özündə conteynirayzd sistemini çalışdıra bilən bir xüsusiyyət kimi başa düşə bilərik.

**Docker Registry**

Bu hissəni isə Docker Host-un özündə Client tərəfindən yazılmış komandaları icra edə bilməsi üçün həmin konteynır və yaxud image-lər üzərində əməliyyat apara bilməsi üçün məhz həmən konteynır və image-ləri Docker Host-a təmin edən hissə kimi başa düşə bilərik. Qısası Docker Registry özündə image-ləri ehtiva edən bir cloud platformasıdır.

**Docker Architecture**

****

Sol tərəf də olan Client hissəsi sizin olduğunuzu bildirir, yəni siz burada Docker texnologiyasını istifadə edən bir client-sınız və terminalda şəkildə göstərilən komandaları yazarkən nəticə etibarı ilə şəklin orta hissəsində yer alan Docker Host ilə əlaqədə olursunuz. Və Docker Host-un içərisində yer alan Docker Daemon siz, yəni Client-ın ona göndərmiş olduğu komandaları icra edən bir program təminatıdır və bu komandaları qəbul edərək container və image-lər üzərində sizin ona göndərmiş olduğunuz container və image-ləri idarə edir. Şəkildən də göründüyü kimi Docker Daemon həm Client və həmçinin də Docker Registery ilə kontakta girir.

**Docker Daemon**

Docker Daemon-u aşağıdakı şəkildə olan ahtapot kimi başa düşə bilərsiniz. Məsələn:



Gördüyümüz kimi ahtapot burada image və konteynırlar ilə məşğul olmaqdadır, yuxarıda biz nə demişdik Docker Daemon Client tərəfindən göndərilən komandaları qəbul edərək məhz həmən komandalar vasitəsində Docker Host-da əməliyyatları icra etmiş olur. Məncə bu şəkildən hərşey sizə daha aydın olar artıq ki, Docker Daemon əslində nədir?

**Docker Basic Terminology**

Docker öyrənərkən burada işlədəcəyimiz bir sıra docker-a aid temrinologiyalar olacaqdır ki, hansıki onları bilmək və anlamaq məcburiyyətindəyik. Onlar hansılardır gəlin biryerdə baxaq.

**Docker Daemon**

Docker Daemon, sizin kompüterinizdə olan əməliyyat sistemi ilə əlaqəyə girərək bir servis və yaxud bir iş görən proses halını alır və o container-i run edə bilir, image-i docker hub-dan yükləyir və s. sizin Client-dan yolladığınız komandalar vasitəsi ilə prosesləri icra edir. Qısası Docker Daemon, Docker Host-da işə düşən və docker obyektlərini yaradıb və onlar üzərində əməliyyatları manipulyasiya edən program təminatıdır. Burada docker obyekt-ləri deyərkən bunları nəzərdə tuturam. Məsələn Images, Containers, Network, Volume, Data və s.

**Docker Client**

Docker Client sizin terminalda yazdığınız komandalardır ki, hansıki Docker Daemon ilə əlaqəyə girir.

**Docker Images**

Docker da Image-ları biz bir fayl kimi başa düşə bilərik. Bəli həqiqətən belədir, onları biz bir fayl kimi başa düşə bilərik. Çünki onlar exe olunmamış bir fayl halında olurlar, hansıki özlərində bizim programımızı işə salacaq konfigurasiyaları(kodlar, library, dependencies və s.) özündə saxlamaqdadırlar və həmçinin Image-lər Docker da container-lərin yaradılmasından ötəri vardırlar. Həmçinin Image-ları biz paket kimi də başa düşə bilərik. Yadınıza gəlirsə mən sizə şəkil göstərdim ki, həmən şəklə biz app-mızın run olunması üçün lazımı gərəkli konfigurasiyaları(dependencies, library və s.) oraya qoyulur və sonrasında bu image-dən container yaradılaraq exe olunmuş formata çevrilir.

**Docker Containers**

Docker da Container-lər Image-lərdən yaranaq ortalığa çıxmaqdadırlar. Biz ilk başda Image-ləri öyrəndiyimizdə orada nə demişdik, demişdik ki, Image-ləri bir paket kimi başa düşə bilərik hansıki app-mıza gərəkli olan bütün konfigurasiya(library,dependencies və s. ) məlumatlarını onun içinə ataraq sonrasında bu image-i çalışdıraraq ondan bir container yəni exe oluna bilən fayl almaqdayıq. Həmçinin Docker da Container-lər app-mızın run oluna bilinməsi üçün gərəkli olan tüm məlumatları özündə saxlaya bilən mövhimat kimi də başa düşə bilərik. Həmçinin Docker da Container-lər biri-birindən isole olunmuş, yəni biri-birindən asılı olmamaq şərti ilə ayrılmış formata və secure dəstəkli bir mövhimat kimi də anlaya bilərik.