Slide 1: Introduction to Docker

დღეს ვისაუბრებთ ერთ ერთ მნიშვნელოვან და ძლიერ ხელსაწყოზე, რომელსაც Docker ეწოდება. დოკერი პლატფორმაა რომელიც საშუალებას გვაძლევს შევკრათ ერთ პაკეტში ჩვენი აპლიკაცია და ის ყველაფერი რაზეც ჩვენი აპლიკაცია დამოკიდებულია. ასეთი კონტეინერი გვაძლევს საშუალებას დარწმუნებულები ვიყოთ, რომ ჩვენი აპლიკაცია ყველანაირ გარემოში ერთნაირად იმუშავებს.

Docker იყენებს ოპერაციული სისტენების დონის ვლირტუალიზაციას კონტეინერების იზოლაციისთვის, რის შედეგადაც თითოეულ კონტეინერში გაშვებულ აპლიკაციას უნიკალური, თავისი, გარემო აქვს.

ოპერაციული სისტემის დონეზე ვირტუალიზაცია aka OS-Level virtualization ტექნოლოგიაა რომელიც საშუალებას იძლევა მრავალი იზოლირებული ინსტანსის გაშვებისა ერთ ფიზიკურ მანქანაზე. ეს ინსტანსები, რომლებსაც კონტეინერებს ვუწოდებთ, იზიარებენ ერთ ოპერაციულ სისტემას კერნელს, თუმცა ერთმანეთისგან დამოუკიდებლად მოქმედებენ. ეს მიდგომა განხვავდება ტრადიციული ვირტუალური მანქანებისაგან, რომლებიც მოითხოვდნენ, რომ მთლიანი ვირტუალკა ამოწეულიყო თავის OS იანად.

Docker კიდევ ერთ მნიშვნელოვან პრობლემას აგვარებს, რომელიცაა „ჩემს მანქანაზე მუშაობს“. გამომდინარე იქიდან, რომ დოკერის კონტეინერი შეიცავს არა მარტო აპლიკაციას არამედ ყველა ამ აპლიკაციის დამოკიდებულებას, კონფიგურაციასა და ბიბლიოთეკას, თუ ჩვენს აპლიკაციას შევამოწმებთ დოკერში კონტეინერში გაშვების შემდეგ, შეგვეძლება ვთქვათ რომ იგი იმუშავებს სატესტო სერვერზეც და პროდზეც. ეს მიდგომა გვიხსნის გარემოს გამოწვეული პრობლემების მეტ წილს.

Docker ის ერთ ერთი მნიშვნელოვანი დადებითი თვისებაა ეფექტურობა. კონტეინერები სისტემის რესურსს უფრო ეფექტურად ყენებენ ვიდრე ტრადიციული ვირტუალკები და მათი რესტარტი/ჩართვა/გამორთვა წამებში შეიძლება. ეს საშუალებას გვაძლევს რომ დანერგვები შევასრულობთ უფრო სწრაფად და მოქნილად.

Docker ის მთავარი კომპონენტი არის Docker Engine, რომელიც კონტეინერებს ამუშავებს. აქვე გვაქვს Docker Hub რომელიც წარმოადგენს დიდ პაკეტების ბიბლიოთეკას. იგი დეველოპერებმა შეკრიბეს და უზრუნველჰყოფს რომ ყველა მსგავსი კონფიგურაცეიბით მუშაობს.