**البداية (عمر):**

اهلا وسهلا فيكم :Pاليوم حابين نحكي عن بعض المشاكل يلي عم نواجها حاليا كمطورين برامج ومدراء نظم

**سلايد 3:**

خلينا نبدا بمشكلة كتيير شائعة واغلبنا بواجهها عند التعامل مع مشاريع الويب الكبيرة يلي مابتكون قائمة بحد ذاتها عل لغة واحدة بل تكون مكونة من عدة مكتبات متداخلة تتواصل بين بعضها وتعتمد عل مكاتب أخرى من اجل ان تعمل واكيد كلنا مرينا متل هيك مشاكل وصعوبات من خلال التوافقية وتعدد النسخ و و و ...

**سلايد 4:**

كان في بعض الحلول يلي كانت قائمة على إضافة أدوات يلي تسمى package manager او مدراء الاعتمادية متل pip بالنسبة لبايثون و gem بالنسبة لروبي npm بالنسبة لـ node.js

حتى انو بعض اللغات راحت ابعد من هيك وعملت بيئة وهمية لتعزل فيها مكتباتها واعتمادياتها

**سلايد 5:**

ولكن بالحقيقة رغم هل شي بقيت المشاكل عالقة في طور تنصيب المشروع على المخدمات deployment وإمكانية توافق هل مكاتب مع بعضها ومشاكل التوسعة والاعتمادية بالنسخ الاحتياطي

**سلايد6:**

من هذه المشاكل بنذكر: 1- كثرة الاعتمادية: يلي بكلفنا وقت كتيير لتنصيب هذه الاعتماديات واحيانن قد تكون هذه الاعتماديات غير موجودة على المخدم مما يضطرنا لنقوم بعملية تجميع الها بشكل يدوي

2- اعتمادية تعتمد بحد ذاتها عل اعتمادية أخرى يلي قد تشترط الاخرى عل وجود مثلا منصة JAVA كاملة ب اصدار معين لها حتى تتوافق معها.

3- تعارض الاعتماديات: حيث قد يتطلب وجود احداها غياب الأخرى او حتى استحالة عملهما معا في نفس الوقت.

4- نسخة الاعتمادية التي تم تنزيلها مغايرة لنسخة التي تم تطوير المشروع بها: والنسخة الجديدة تاتي بتغيرات غير متوافقة رجعيا

**سلايد 7:**

كل هذه المشاكل وغيرها بتجعل من إمكانية نقل المشروع من مكان الى اخر او من مخدم ل اخر عملية شاقة جدا ومكلفة.

ويلي بتشكلي هاد الشكل من مايدعى ب dependence hell matrix

**سلايد 8:**

إضافة لكل يلي ذكرناه سابقا في عنا مشاكل أخرى يلي بنقدر ندرجها بصعوبات النشر والتنصيب ونقل المشاريع

يلي هية حتكون :

* بتتعلق ب متقيرات البيئة الجديدة يلي يجب علينا تعينها مسبقا
* اعداد نظام ال Cash
* اعدادات الأمنية عل الخادم
* إضافة الى تحديث نظام الخادم وبرمجياته ويلي قد تسبب عطلا في المشروع

**سلايد 9:**

كل هذه المشاكل بتسبب تأخير وتعطيل بالعمل إضافة الى وجع الراس لساعات طويلة :P

**سلايد10:**

هون اجا الحل بشي يدعى ال Docker

ال دوكر هو عبارة عن بيئة عمل خاصة بالمطورين ومدراء النظم بيسهل عملية تطوير التطبيقات ونقلها وتثبيتها في اسرع وقت ممكن

**سلايد 11:**

ال دوكر هو طبقة عل مستوى نظام التشغيل بتسمحلنا بعزل المهام والعمليات Processes وعمليات الادخال والإخراج I/O وحجز الذاكرة وغيرها

حيث يقوم ب انشاء حاويات او كونتينر حيث ان هذه الحاويات تلعب دور غلاف حاوي للتطبيق بحيث يصبح قائما بحد ذاته ومكتف ذاتيا

أي ان هذه الكونتينر بتحوي المشروع إضافة الى كل الاعتماديات الخاصة به التي يحتاجها للعمل مع توزيعة المناسبة مثل Ubuntu او او الخ بجميع هيئتها و متغيرات البيئة التي يحتاجها

**سلايد 12:**

لتقريب الصورة ف نحنا ب استعمال ال Docker بقدر

طور تطبيقي عل Ubuntu مثلا عل نسخة معينة مثلا 14.02

وتطبيقي يحتاج الى مكتبة python بالتحديد نسخة 7.02 ويستخدم مكتبة Django  مع داتا بيز MySQL وسيرفر apache

ف اعطي الامر ل docker ان يقوم ب انشاء container حاوية بهذه النسخة من التشغيل مع يقوم بتثبيت نسخة 2.07 python ويقوم ب استدعاء مدير حزم الملفات pip لتثبيت مكتبات ال python المتوافقة ومن ثم يتم جمع هذا كله في image واحدة يمكن تشغيلها عل أي خادوم او حاسوب به docker

**الفقرة Containers vs VMs (محمد):**

**شريحة VMs vs. Containers**

هلأ قبل ما نحكي الفرق بين الكونتانير والفي أم وير خلينا نحكي الشي المشترك بيناتهن، عملياً التنتين ألهن نفس الهدف يلي هو عزل التطبيق أو النظام ونخليه يشتغل بوحدة مستقلة فينا نشغلها وين ما بدنا، والتنتين عملو Abstraction وريحونا من هم ال Physical Hardware وصرنا عم نشتغل بطبقة فوق الفيزيكال كلو هاد أدى إلى استخدام الموارد بكفاءة عالية.  
هلأ الفرق بيناتهن هو من الناحية المعمارية architectural.

**شريحة VMs**

VMs متل ما منعرفها هي محاكاة لكومبيوتر حقيقي، وبتشتغل من خلال شي اسمو Hypervisor وأشهر مثال عليه هو ال Vmware و Virtual Box يلي كلياتنا منستخدمهن ومنزلينهن على أجهزتنا، لما منزل Hypervisor مثل فرتشوال بوكس مثلاً على جهازنا هون بيصير اسمو Host Machine وهو بيعطي جزء من مواردو (رام، CPU) لل Hypervisoer وهي ببتولى مهمة توزيعها وادارتها بين VMs يلي منسميهن Guest Machine.

ومتل ما منعرف كل VM أو GM هي بحد ذاتها نظام تشغيل مستقل بكلشي يعني ألها System Bins و Libraries و Network Adapter Virtualالخاص فيها، فاذا بدنا نشوفها من الداخل منشوفها كوحدة مستقلة ألها مواردها الخاصة، أما من الخارج فنحنا منعرف أنو مجموعة هل GMs كلهن عم يتشاركو الموارد تبع Host Machine.

ففينا نشبه ال VM ب House متل يلي بالصورة او نحنا منسميها فيلا، فالهاوس متل ما منعرف بيكون كلشي فيها مستقل بالبنية التحتية كالتدفئة والكهرباء والمي..الخ، فكل الهاوسات بتلاقي فيها كحد أدنى غرفة نوم وغرفة معيشة وحمام ومطبخ فلما الواحد بدو يشتري هاوس على طول بيشتري شي أكثر من حاجتو بدون ما يستفيد منو، ونفس الشي بالنسبة لل VM

ومتل لما بدنا هاوس جديد فنحنا مضطرين نبني واحد من الصفر كمان ال VM لما بدنا وحدة جديدة فلازم نعمل deploy لنسخة كاملة منها بما تحويها من نظام تشغيل وبرامج يعني كلشي تقريباً.

وال VM فيها هدر للموارد لأنو من أجل كل instance عندي نسخة كاملة من نظام التشغيل.

مما سبق نستنتج أنو VMs قدمتلي Hardware Virtualization.

**شريحة Containers**

الكونتاينر بتأمنلي Virtulizatoin على مستوى نظام التشغيل، يعني عندي مجموعة من الكونتاينرز بتتشارك ال Kernal تبع نظام التشغيل المضيف، ومتل مو شايفين بالصورة كل كونتاينر بتحوي فقط lib و bin يلي بتحتاجها بعلمها بحسب الأنظمة او البرامج يلي فيها وبتتشارك بنفس الوقت مع غيرها بل Kernal تبع نظام التشغيل.

ففينا نشبها بالشقة Appartment فالشقق بالبناية كلها بتتشارك بالبنية التحتية متل التدفئة المركزية، والشقق عادة بتجي بحجموم وقياسات متعددة فالواحد بيحسن يلاقي شقة على كيفو بحسب حاجتو ونفس الشي بالنسبة للكونتاينر.

**شريحة They’re different**

طيب السؤال يلي بيخطر على البال هل يمكن الدمج بين ال VM و الكونتاينرز ؟

أي أكيد فينا، يعني فينا نشغل ضمن ال VM نحط ما يسمى Conaitner Engine او لح نشوف لاحقا Docker Engine ونشغل عليه العديد من الكونتاينرز فما فرق فينا بالنسبة للوكنتاينر بتشتغل وين ما كان.

متل مو شايفين بالصورة يلي على اليسار هون حاطط ال Engine مباشرة على نظام التشغيل وهاد الانجن بقلبو أكتر من كونتينر، بينما بالصورة يلي على اليمين هون حاطط ال VMs مباشرة على الهاردوير بدون نظام تشغيل وبقلب أول VM حاطط ال Engine وبقلبو كونتاينر او اكثر.

**الفقرة What & Why (باسل):**

**سلايد 20 - What is Docker**

شو هو الدوكر؟ الدكور عبارة عن عن بيئة عمل اوبن سورس خاصة بالمطورين ومديري الانظمة مكتوب بلغة Go بيعتمد على لينوكس كونتاينرز والغرض الأساسي منو هو تسهيل عملية تطوير التطبيقات والنظم داخل الكونتاينرز يلي شفنا مبدئها من شوي، وهو لحتى يشتغل بيستخدم ميزات أساسية للينكس كيرنال منها namespace و control groups لحتى يحسن يعمل كونتاينرز على نظام التشغيل العادي، فهو يقوم بإنشاء حاويات (containers)، بحيث أن هذه الحاويات تلعب دورغلاف حاوي للتطبيق، بحيث يصبح قائما بذاته، مكتفي ذاتياً.

الشي الأساسي يلي عملو دوكر هي تسهيل التعامل مع الكونتاينرز فباستخدامو صار أي حدا سواء كان مطور او مدير نظام او حتى مصمم نظام بيحسن يساوي كونتاينر بسهولة ويجربو عليها التطبيقات يلي عم يشتغلو عليها، والتعامل معو سهل لدرجة أنو أي حدا منكن بكم سطر كود فيو يحط برنامجو بقلب كونتاينر مع الحبشات يلي بيعتازها البرامج وبتصير هي الحاوية بتشتغل بدون تعديل عندو رفيقو او على سيرفرات غوغل او سيرفرات امازون.

حتى صار شعارو هو Build Ship Run Any App Anywhere شو يعني هل الشي ؟ يعني المطورين صارو بيحسنو يختارو اللغة أو Compunets يلي بدهن يبنو نظامهن فيها سواء كانت Java دونت نيت الخ، ويتأكدو أنو شو ما كودو وعملو بالاخير لح يشتغل مهما كانت بيئة العمل الفعلية يلي لح يختارها Ops سواء اختارو بيئة فيزائية او فيرتشوال او كلاود.

وهون صار فيو الواحد ينتقل التطبيق او النظام بسهولة بين المراحل عمر المشروع يلي منعرفها (تطوير ، اختيار ، تشغيل) مع التأكد انو كل مرة عم تكون نفس بيئة العمل شو ما كانت المرحلة.

**سلايد 22- Why Developer Care**

بعض الفوائد يلي بيجنيها المطورين من استخدام ال docker:

* القدرة على إنشاء نسخة من الأبلكيشن متكاملة وقابلة للتشغيل بأي وقت وعلى أي جهاز.
* وبالتالي ما عاد آكل هم مشكلة الاعتمادايات وعدم التوافق لما بشغل الأبلكيشن على حواسييب وأنظمة أخرى.
* كل تطبيق بقدر شغلة ب container معزولة وبالتالي صار فيني مع كل كونتينر مثلا جرب نسخة مختلفة من المكتبة.
* أتمتة الاختبارات
* تقليص والقضاء على مشاكل عدم التوافقية.

**سلايد 23- Why Devops Care**

أما عن بعض الفوائد يلي لح يجينها المطورين ومسؤولي العمليات اذا استخدمو دوكر فهي:

* جعل دورة حياة المشروع أكثر تناسق.
* تحسين نوعية الرماز الذي بيكتبه المطورين.
* القضاء على التناقضات بين بيئات التطوير، الاختبار، الاستخدام عند الزبائن.
* دعم الفصل بين الواجبات.
* تحسن كبير في سرعة وموثوقية نشر المستمر وتكامل النظام المستمر.
* الكلفة أقل من vm.. وما منسى انها اوبن سورس

**سلايد 25**- **separation of concerns**

متل مانا شايفين هون في عنا كونتينر حقيقية متل الموجودة بالمواني، بكل كونتينر المسؤول عن الأخياء الخارجية عم بكون مسؤول العمليات بنظامنا متل عمليات المراقبة واعدادات الشبكة وغيرها، أما المسؤؤول عن الأمور داخل الكونتينر فهم المطورين أنفسهم من كود ومكاتب ممكن يعتمد عليها وداتا وغيره.

**سلايد 27**

من أحد أهم التطبيقات للدوكر هو عملية التيستتنغ فإذا بدي اعمل تستنغ لأبلكيشن ما باستخدام الفيرتشوال ماشين واللي بتستخدم مبدأ الfull clone فعملية الاقلاع رح تكلفني عدة دقائق، هون طبعاً لازم نذكر انو عدد النسخ يلي بقدر أقلها من الفرتشوال ماشين محدودة. اطلاقاً من هل حقائق لحتى ألع ألف نسخة اختبار هاد الشي رح ياخد مني ساعات إذا مو أسابيع ولحتى تتم كامل عملية الاختبار رح ياخد هاد الشي أيام إذا مو ساعات.

بالوقت يلي بأدر لنفس الأبلكيشن شغله بأقل من نص ثانية مع إمكانية تشغيل 1000 نسخة اختبار وبالتالي صار كامل عملية الاختبار بتاخد معي ساعات بدل من أيام أو أسابيع.

**سلايد 29- Docker components (آلاء):**

هلأ رح نعمل جولة سريعة عن المكونات الأساسية يلي بتشكل الـ Docker

Image: وهية المكون الأساسي بالدوكر. بالحقيقة هي الأبلكيشن كاامل مع كل المكاتب أو الأشياء اللي ممكن يستخدمها هل الأبلكيشن أو يعتمد عليها بحيث يصير يقدر هل برنامج يشتغل عأي جهاز دور الحاجة لتنصيب أي شي آخر. وهية نفس فكرة ملفات الـ iso

بتم خلق هل امج باستخدام Dockerfile

Container: هي المكان يلي بحتضن الأمج واللي بيتم تنفيذ الأمج ضمنها وبالتالي هية البيئة التنفيذية. بالحقيقة فينا نقول انو الكونتينر هو runtime instance من الأمج.من الممكن أن يكون التطبيق عبارة عن حاوية واحده او عدة حاويات تتواصل فيما بينها تحتوي علي كل ما يحتاجه المشروع

Engine: هو المسؤول عن خلق وإنشاء الكونتينرات بدء من إمج.

Registry Service: مهمتها التحكم بتخزين وتوزيع الأمجات.

طيب شو الفرق بين الأمج والكونتينر:

الأمج ما بيتم التعديل عليه بتاتاً، بعكس الكونتينر.

بنية الأمج بتكون بحيث كل ما منضيف فانكشناليتي جديدة علأمج بتنضاف طبقة read only جديدة وهي الطبقات غير قابلة للتعديل

أما لما منعمل new instance من container فالاضافة لبطقات الكونينر الأصلية بتنخلق طبقة read /write layer وبتنضاف على سطح طبقات الأمج الغير قابلة للتعديل. وبالتالي أي تغيير بيطرأ على الكونينر أثناء التنفيذ، مثلا عملية كتابة بملف اللوغ فبينعكس هاد الشي على طبقة ال read/write يلي انضافت مؤخراً أما الطبقات الأخرى فمتل ما حكينا ما بيصير علها أي تعديل. طبعا لما بعمل exit ل container ما بفقد المعلومات يلي قمت بكتابتها إلا بحالة حذف هي الكونتينر.

وبالتالي صار فينا نشبه الأمج بال class والكونتينر بكون object أو instance من هل class

استخدام طبقات الكونتينر بيوفر كتير ميزات منها السرعة وانقاص مسحة التخرين المطلوبة، فلما منخلق كونتينر جديدة ما بيتم نسخ كامل طبقات الأمج وانما بيخلق طبقات الكونتينر وبيربطها مع طبقات الأمج الموجودة. وهذا الشي بيخلقلي سرعة كبيرة.

-ميزة كتير مهمة بالدوكر انه مو بس بيتم التشارك بنفس الصورة بين عدة أمجات وانما حتى بتم التشارك بنفس الطبقات بين عدة أمجات. فمثلاً إذا شركة بستخدم نظام تشغيل معين مع كل الامجات بالتالي بتكون طبقة نظام التشغيل مشتركة. فمتل ما عم شنفو بالصورة اذا بدنا نخلق امجين وحدة php والتانية Node.js بس التنين بيعتنمدوا عنفس نظام التشغيل بتم خلق طبقة نظام تشغيل وحدة وعليها بينبى أمجين بطقات مختلفة حسب الوظيفة يلي بدها تأديها كل أمج.

* أما عن إمكانية إنشاء عدة نسخ من الكونتيتنر فتم عن طريق ال docker compose. فببساطة منحط بال Compose file الكونفغريشنز لكل السيرفسز وبعدين بتعملية وحدة بتم خلق وإقلاع كافة الكونتينرات بدء من الملف المعرف سابقاً.
* نتيجة لإمكانية إيجاد مجموعة كبيرة من الكونتينرات ظهرت مشكلة إدارة هل كونتينرات وخاصة لما بتكون على هوست واحد.. لهيك تم إيجاد شي بيقدر يدير هل كونتينرات اسمه docker swarm من خلاله بيقدر المستخدم يعنقد أو يجمع مجموعة من العقد بالدوكر وبالتالي بتصير كسيتم افتراضي واحد. كما انه بيعمل على توزيع الور لود وبالتالي بيخففلي التعقيد.